

Order No. 002547 Revision 03 Software Release 9.0

DOMAIN System Command Reference

DOMAIN System Command Reference

Order No. 002547 Revision 03 Software Release 9.0

Apollo Computer Inc. 330 Billerica Road Chelmsford, MA 01824 Copyright © 1985 Apollo Computer Inc.

All rights reserved.

Printed in U.S.A.

First Printing:

May, 1983

Latest Printing:

July, 1985

Updated:

July, 1985

This document was produced using the SCRIBE® document preparation system. (SCRIBE is a registered trademark of Unilogic, Ltd.)

APOLLO and DOMAIN are registered trademarks of Apollo Computer Inc.

AEGIS, DOMAIN/IX, DOMAIN/Dialogue, D3M, DPSS, DGR, GMR, GPR, and DSEE are trademarks of Apollo Computer Inc.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF APOLLO COMPUTER INC. HARDWARE PRODUCTS AND THE LICENSING OF APOLLO COMPUTER INC. SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE—TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY APOLLO COMPUTER INC. FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY BY APOLLO COMPUTER INC. WHATSOEVER.

IN NO EVENT SHALL APOLLO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATING TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF APOLLO COMPUTER INC. HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

Preface

The DOMAIN System Command Reference is the third volume in the three-volume introduction to the DOMAIN Computing System. The first volume, Getting Started With Your DOMAIN System, provides a tutorial approach to getting started on your node. The second volume, DOMAIN System User's Guide, constitutes a handbook that takes you beyond the introductory stage into practical applications of Display Manager (DM) and Shell operations. This third document provides complete reference information on all of the DM and Shell commands that are available to you. We assume that you are familiar with the material in the first two books before you attempt to use this reference manual. Fundamental concepts like file structure and usage are taken for granted here. We tell you how to use the commands; not why you might want to use them.

The organization of this manual has changed substantially from Revision 02 to Revision 03. All commands are now ordered alphabetically in two major chapters (one each for the DM and the Shell). Fundamental constructs (such as DM regular expression and Shell wildcard syntax) are summarized in the introductory chapters that precede the command descriptions.

The special DM and Shell index pages which classify groups of related commands remain as an aid to locating the commands that you need.

Appendix E, "Node Recovery Procedures", in Revision 02 is no longer in this manual. Individual Owner's Guides are now available for each node type; refer to those books for information about recovering from node hangs and crashes.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the description of the Shell command CRUCR (CREATE_USER_CHANGE_REQUEST). You can also get more information by typing:

\$ HELP CRUCR <RETURN>

For your comments on documentation, a Reader's Response form is located at the back of this Guide.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions. **UPPERCASE** Uppercase words or characters in formats and command descriptions represent commands or keywords that you must use literally. lowercase Lowercase words or characters in formats and command descriptions represent values that you must supply. Square brackets enclose optional items in formats and command descriptions. In sample Pascal statements, square brackets assume their Pascal meanings. } Braces enclose a list from which you must choose an item in formats and command descriptions. In sample Pascal statements, braces assume their Pascal meanings. A vertical bar separates items in a list of choices. Angle brackets enclose the name of a key on the keyboard. CTRL/Z The notation CTRL/ followed by the name of a key indicates a control character sequence. You should hold down the <CTRL> key while typing the character. Horizontal ellipsis points indicate that the preceding item may be repeated one or more times. Vertical ellipsis points mean that irrelevant parts of a figure or example have

been omitted.

Summary of Technical Changes

Display Manager

The following new features are available in the SR9.0 Display Manager.

New Commands

- ENV (ENVIRONMENT): Set or display an environment variable.
- TNI (TO NEXT ICON): Move cursor to next icon.

Changes to Existing Commands

- LO [-ON|-OFF]: Enable/disable logoff capability.
- SQ (SEARCH_QUIT): Now identical to ABRT.
- WP (WINDOW POP): Now works with named windows and window groups.

Shell

The following new features are available in the SR9.0 Shell.

New Commands and Features

- Active functions: You may now use the construct `"command" to assign the value returned by 'command' to a Shell variable. See the DOMAIN System User's Guide for details.
- DSPST (DISPLAY_PROCESS_STATUS): Display process status graphically.
- EDNS (EDIT NAMING SERVER): Invoke naming server editor.
- EXPORT: Change a Shell variable into an Environment variable.
- FOR: Execute a FOR statement.
- HLPVER (HELP_VERSION): Provide HELP support for Shell scripts.
- LTY (LIST_TYPES): List installed types.
- NOT: Negate a Boolean value.
- OLD _EDFONT: Former version of EDFONT with function key interface.
- PRF (PRINT_FILE): Now supports Imagen 8/300 and Imagen LBP-10 laser printers connected via SIO lines or the Multibus.
- PRFD (PRF_DISPLAY): Invoke menu-based PRF.
- SELECT: Execute a SELECT statement.
- SEND_ALARM: Send messages to alarm servers.
- SET: Set current Shell conditions.
- SOURCE: Execute a Shell script at the current Shell level.
- VSIZE (VT100_SIZE): Set or display VT100 window settings.

Changes to Existing Commands

- ARGS -ERROUT: Now writes output to error output if desired.
- BIND: Several new options: -ALLRESOLVED, -BDIR, -LOCALSEARCH, -NOLOCALSEARCH, -MULTIRES, -NOMULTIRES, -MAKERS, -NUND, -SET_VERSION, -SYSTYPE.
- CPBOOT: Now supports cartridge tapes.

- CPF -DU: Now deletes currently locked files once they are unlocked.
- CPT -F: Force tree deletion is you have P rights.
- CRP: New option: -ME; new functionality in -LOGIN; -MBX no longer available.
- CRUCR: New corporate address for UCR submissions.
- CTNODE: Several new options: -FROM, -ON, -MS, -MD, -IDUPL, -LC, -ROOT, plus three new paragraphs in the description explaining "confusion resolution".
- DEBUG: New information on macros and variables, new option, -PROC, for cross-process debugging
- EDACL: New option, -UNIX, plus new directory rights "S" (search) and "E" (expunge).
- EDFONT: New version with a pointing device-driven interface.
- EDMTDESC: Now supports cartridge tapes.
- EMHASP: No longer standard software. Documentation is now provided with the RJE product.
- EXISTF: Wildcarding in pathname arguments is now supported.
- EXIT: Now works for FOR and SELECT as well as WHILE.
- FPAT: New options: -RM, -RMF, -LM.
- HASPSVR: No longer standard software. Documentation is now provided with the RJE product.
- INLIB: Now supports multiple pathnames and wildcarding.
- INVOL: Default paging size (option 8) is now 352 pages instead of 256. A new option 10 allows you to set or display the sector interleave factor for a volume. (See Appendix D.)
- LAS: New options: -ALL, -FROM, -TO, -PROCESS.
- LCNODE: New options: -C, -ID.
- LD -ROOT: Supports new naming server.
- NETSVC -S: Controls network server availability on a node.
- NEXT: Now works for FOR and SELECT as well as WHILE.
- PRF: Completely overhauled for SR9.
- PROBENET: New options: -D, -SENS. Deleted option: -Q. Comment character changed from '(blank)' to '#'. Reported statistics also changed.
- PRSVR -N: Assign a name to the print server process.
- RBAK: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN, -REWIND. Limited floppy disk support.
- READ: New option, -ERRIN, for reading from error input; new -TYPE choices: ANY and ENV.
- READC: New option, -ERRIN, for reading from error input.
- READLN: New option, -ERRIN, for reading from error input.
- RETURN -OK: Same as -TRUE.
- RWMT: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN. Limited floppy disk support.
- SALVOL: Now allows controller unit number specification for use when two or more of the same storage device are attached to one node.
- SH: Completely overhauled for SR9.
- TCTL: Several new options: -[NO]RTS ENABLE, -FORCE, -NO[RAW].
- WBAK: Now supports cartridge tapes with the options -DEV CT, -RETEN, -NRETEN, -SYSBOOT, -NO_EOT. Limited floppy disk support.

NOTE: A vertical bar in the margin indicates a substantive technical change from Revision 02.

Contents

Chapter 1 Display Manager Basics	1-1
1.1. Defining Points and Regions	1-1
1.2. Defining Window Boundaries	1-3
1.3. Defining a Range of Text	1-4
1.4. Using Regular Expressions	1-4
1.5. Key Naming Conventions	1-8
1.5.1. Standard Key Names	1-8
1.5.2. Controlling Keys from Within a Program	1-9
1.6. Special Characters in DM Scripts and Key Definitions	1-11
1.0. Special Characters in DW Scripts and Rey Definitions	
Chapter 2 Display Manager Commands	2-1
Lisbai	9-1
Chapter 3 Shell Basics	3-1
3.1. Command Format	3-1
3.1.1. Arguments	3-1
3.1.2. Separators	3-2
3.2. Using Special Characters	3-2
3.3. The Command Line Parser	3-3
3.3.1. Standard Command Options	3-5
3.3.2. Command Line Parser Options	3-5
5.5.2. Command time I arser options	
Chapter 4 Shell Commands	4-1
Appendix A CALENDAR	A-1
Appendix B OLD_EDFONT	B-1
B.1. Background	B-1
	B-3
B.2. Overview	B-4
B.3. A Brief Walk-Through	B-6
B.4. Detailed Menu Descriptions	D-0
Appendix C EDFONT	C-1
C.1. Introduction	C-1
C.1. Introduction	C-2
C.2. Invoking EDFONT	C-3
C.3. Sample EDFONT Display	C-9
C.4. Using EDFONT	0-9

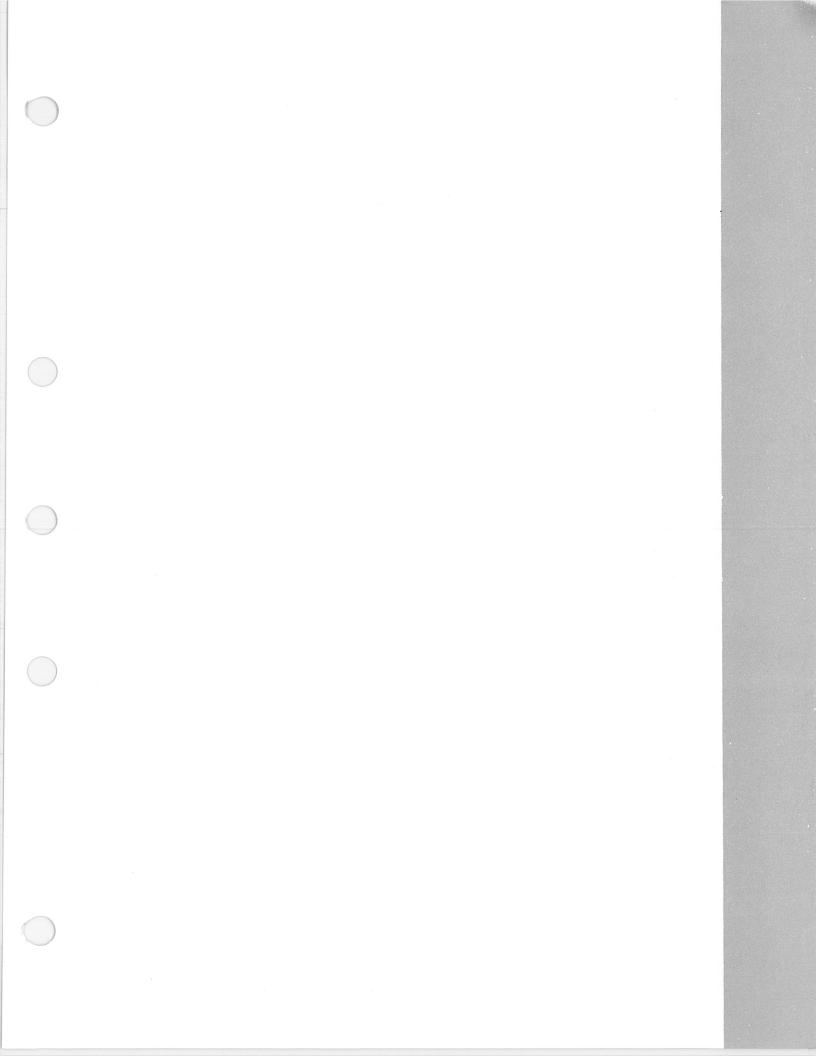
D.1. Background D.2. Invoking INVOL D.3. Operations Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations D-1 D-1 D-1 D-1 D-1 D-1 D-1 D-1 D-1 D-	C.5. Glossary of Terms	C-17
D.1. Background D.2. Invoking INVOL D.3. Operations Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations D-1 D-2 E-1 E-1 E-1 E-1 E-1 E-1 E-1 E-1 E-1 E-		
D.2. Invoking INVOL D.3. Operations Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E.5. D.1 D.1 D.1 D.2 E.1	Appendix D INVOL	D-1
D.2. Invoking INVOL D.3. Operations Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E.5. D.1 D.1 D.1 D.2 E.1		
D.3. Operations D-2 Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E.5 E.6		D-1
Appendix E SALVOL E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-1 E-1 E-1 E-2 E-3 E-3 E-3 E-6	•	D-1
E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-6	D.3. Operations	D-2
E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-6		
E.1. Background E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-6		
E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-6	Appendix E SALVOL	E-1
E.2. Invoking SALVOL E.3. Salvaging Strategy E.4. Limitations E-6	E.1. Background	D 1
E.3. Salvaging Strategy E.4. Limitations E-6		
E.4. Limitations E-6	•	
Index	D.I. Dimivacions	E-0
Index		
	Index	Indov-1

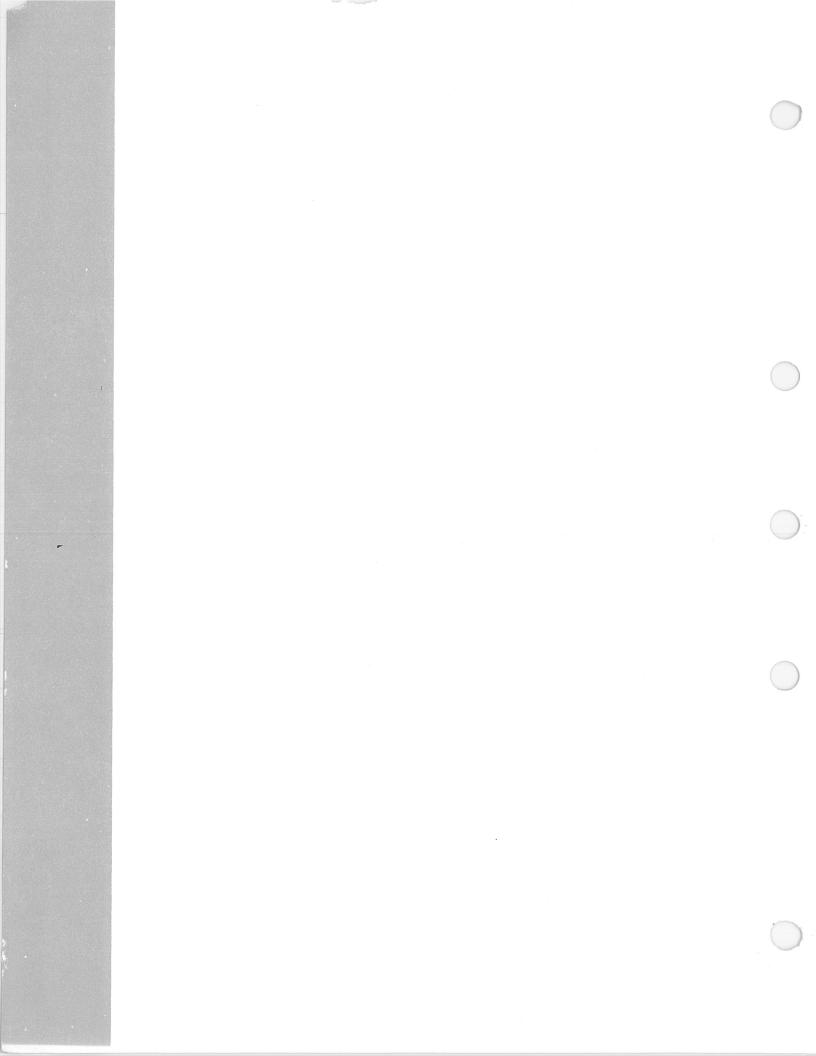
Illustrations

Figure 1-1.	The Low-Profile Keyboard Map	1-10
Figure 1-2.	The 880 Keyboard Map	1-10
Figure 3-1.	Typical Shell Command Format	3-1
Figure 4-1.	An Access Control List (ACL) with Two Entries	4-91
Figure 4-2.	Applying Initial ACLs	4-95
Figure C-1.	Sample EDFONT Display	C-4
Figure D-1.	Sample Logical Disk Organizations	D-2

Tables

Table 3-1.	Command Shell Special Characters	3-3
Table 4-1.	Access Rights for Files and Directories	4-93
Table 4-2.	Abbreviations for Commonly Assigned Rights	4-94
Table A-1.	Valid Time Zones	A-2





Chapter 1 Display Manager Basics

This chapter summarizes the basic concepts that apply to the DM commands described individually in the following chapter. For a more indepth discussion of these concepts, please refer to the DOMAIN System User's Guide.

1.1. Defining Points and Regions

Unless otherwise noted, all DM commands must be preceded by a pointing operation. This generally involves moving the cursor to the spot where the command is to be executed (for example, pointing to the window that you want to scroll), or specifying a specific screen or line location as a command argument. If you don't specify some pointing function, the Display Manager executes the command at the current cursor position.

To point, simply move the cursor to the desired location. For example, to point to a window, place the cursor anywhere inside the window. The command reads the cursor position to determine which window you mean. (Please note that when you use the block cursor to designate a point on the screen, the designated point is actually at the lower left corner of the block cursor.)

You can also define a point in any of the following ways:

line-number

Line numbers begin at 1 and range upward to the last line in the pad. There is no fixed limit to the number of lines a pad can contain. You may use the symbol "\$" to refer to the last line in the pad. Remember that the edit pad window legend contains the line number of the top line in the window for reference. You may also display the line number (plus the column number and X and Y coordinates) of the current cursor position by using the DM command "=".

+/- line-number

The "+/- line-number" format denotes the nth line before or after the current cursor position in a pad.

[[line-number],[column-number]]

This format indicates the point by line and column number in the pad. The Display Manager assumes the current line if the first portion is omitted, and column one if the second portion is omitted. Line numbers range from 1 to the last line in the pad (no limit). Column numbers range from 1 to 256. When you specify a point in this format, you must use the outer set of square brackets to enclose the numbers. This is how the Display Manager distinguishes between line/column positions in a pad and X/Y coordinates on the screen (below). Note that the use of "\$" to denote the last line in the pad does not work within square brackets.

Examples: [127,14] Line 127, column 14

[53] Line 53, column 1. Brackets are optional in this case: see above.

[,12] Column 12 of the current line

([x-coordinate],[y-coordinate])

Screen coordinates specify bit positions on the display. The origin (0,0) is at the extreme upper left corner of the screen. Values for x-coordinates range from 0 to 799 (portrait) or 0 to 1023 (landscape), and values for y-coordinates range from 0 to 1023 (portrait) or 0 to 799 (landscape). The current x or y coordinate of the cursor is used if you omit it from the coordinate pair. When you specify a point in this format, you must use the outer set of parentheses to enclose the numbers. This is how the Display Manager distinguishes between line/column positions in a pad (above) and X/Y coordinates on the screen.

Examples: (200,450) Bit position with the given coordinates

(135) Bit position whose X coordinate is 135 and whose Y coordinate is the same as

the current cursor position.

(,730) Bit position whose X coordinate is the same as the current cursor position and whose Y coordinate is 730.

/regular-expression/ or \regular-expression\

A regular expression specifies a string in the pad that begins or ends the region of interest. Regular expressions are described in section 1.4.

Now that we can identify points, let's turn to regions. A **region** is simply the area between two points. To define a region, you need to use the DM command DR (DEFINE_REGION). The region definition operation has the following format:

[point] DR; [point]

The first point marks one corner of the region. The second point marks the opposite corner of the region. Remember that the points may be defined by cursor positions or specified explicitly in one of the alternate formats mentioned above.

For convenience, the predefined key, <MARK>, invokes the DR command. Point the cursor at the start of the range, MARK it, then point at the end of the range. That's all there is to it.

For those DM commands that require you to specify a region in which to operate, you can declare it either by MARKing it, or by explicitly specifying the region with one of the techniques described above. If a DM command does require you to define a region, the command is specified in the following format:

[region] command

The symbol [region] indicates where you must define the region. (The definition of a range for text editing operations -- cut, paste, substitute, etc.-- is slightly different. Refer to section 1.3 for more information.)

1.2. Defining Window Boundaries

When a window's size or position on the screen is changed in any way, the DM determines the new boundaries of the window using calculations based on a pair of points (a "point pair") on the screen. Usually, the first point in the pair has been defined with the DR command and the second point is the current cursor position, although you may provide absolute point coordinates as described in section 1.1.

Each point may specify either a new or existing edge of a window, or a new or existing corner of a window. The new window, then, is created based on the relationship between the X and Y coordinates of the two points. When either of the points specifies a new upper edge or right edge for a window, the position is adjusted to account for the size of the displayed block cursor, because the actual coordinates of the cursor are determined by its lower left corner. This adjustment is made only when the coordinate source is the block cursor, and not when the point comes from the touchpad or mouse, or from explicitly entered coordinates.

The relationship between the two points in the point pair affects the actions of the window-related commands CP, CE, CV, CC, WG, WM, and WDF in the following ways:

1. Horizontal movement only (y coordinates of the two points are equal):

Creation - Create a window bounded by the given x coordinates, the top of the screen, and just above the normal DM command window (i.e., a full vertical window).

WG/WM - Select the unobscured vertical edge nearest to the first point, and change the x coordinate of that edge to be that of the second point. The y coordinate of the first point must be within the unobscured range of y coordinates of the selected edge.

2. Vertical movement only (x coordinates of the two points are equal):

Analogous to horizontal movement, except that for creation, the full horizontal width of the screen is used.

3. No movement (two points are equal):

Creation - create a 512 by 512 window centered as nearly as possible (subject to screen boundaries) on the given cursor position.

WG - treated as in 4 below.

WM - Select the unobscured corner nearest the given point, and move the corner to that point.

4. Two points differ in both x and y:

Creation - The given four coordinate values form opposing corners of the window.

WG/WM - The first point selects the nearest unobscured corner (the corner itself must be visible) and that corner is repositioned at the second point.

If only one point is given (i.e., the DR command is not issued), grow is illegal and move behaves as in step 3 above. The DM uses one of its five default window regions, or a default determined by the last window creation or deletion (WC) command, as follows:

- If the last such command was window deletion (i.e., WC), the default region is the same as that of the deleted window.
- If the last such command was a successful window creation command, the default region is the next third of the screen following the created window.
- If the last such command was an unsuccessful window creation command, the default region is the same as specified in that command.

To define the five default window regions, use the DM command WDF.

1.3. Defining a Range of Text

The text editing commands that perform cut, paste, and substitute functions operate on a range of text. That range is declared just as you would mark any other region in a pad; i.e., you place the cursor at the start of the range, press <MARK>, then move the cursor to the end of the range and issue the command in question.

The region of text you define for a cut, paste, or search operation is highlighted in reverse video when you use the <MARK> key. This is because that key invokes the DR;ECHO command sequence. You can still use the DR command alone to place a mark, but the highlighting feature is not invoked without ECHO. You can cancel the defined ranged with the ABRT command (invoked with CTRL/X). Refer to the descriptions of the DR, ECHO, and ABRT commands for more information.

Please note that the character under the cursor at the end of the range is NOT included within the range. Note also that you may NOT declare a range explicitly as an argument to the editing commands, since those commands do not, in general, accept arguments. You must use the MARK key or the DR command sequence.

The default range is different for these editing operations, too. Whereas the general DM default range is the current cursor position; cut, paste, and substitute commands apply to all characters from the current cursor position up to the end of the line (including the NEWLINE character) if no other range has been marked immediately prior to invoking the command.

1.4. Using Regular Expressions

Special "regular expression" notation is used to specify patterns for search and substitute strings in the Display Manager editor. This notation is also used in the Shell commands ED (EDIT), EDSTR (EDIT_STREAM), FPAT (FIND_PATTERN), FPATB (FIND_PATTERN_BLOCK), and CHPAT (CHANGE_PATTERN). Regular expressions permit you to concisely describe textual patterns without necessarily knowing their exact contents or format. You can create expressions to describe patterns in particular positions on a line, patterns that always contain certain characters and sometimes include others, or patterns that match text of indefinite length.

Regular expressions are constructed as follows:

1. Any standard ASCII character (except those discussed below) is a regular expression and matches one and only one occurrence of that character. (For multiple occurrence

matches, see "*" below.) The case of the characters in the expression is not significant by default. Use the DM command SC (SET_CASE) to control case significance.

SAM

fred12

All valid expressions.

Joe (a&b)

2. A percent sign (%) at the beginning of a regular expression matches the empty string at the beginning of a line. If a "%" is not the first character in the expression, then it simply matches the percent character. Use this special feature to mark the start of a line in a regular expression.

"%Print" matches the string in line a but not line b, because in line b Print is not at the beginning of the line.

- (a) Print this file.
- (b) This Print file.
- 3. A dollar sign (\$) at the end of a regular expression matches the null character at the end of a line. If "\$" is not the last character in the expression, then it simply matches the dollar sign character. Use this special feature to mark the end of a line in a regular expression.

"file\$" matches the string in line a but not line b, because in line b file is not followed by an end-of-line marker.

- (a) Print this file
- (b) This file is permanent
- 4. A question mark (?) matches any single character except a NEWLINE character. The only exception to this is if the "?" appears inside a character class (see below), in which case it represents the question mark character itself.

"!OLD!!!" matches a and b, but not c, because in line c the letters "OLD" are alone on the line.

- (a) HOLDING
- (b) FOLDERS
- (c) OLD
- 5. Whereas the above expression (?) matches only a single occurrence of a pattern, an asterisk (*) following a regular expression causes it to match zero or more occurrences of that expression. The only exception to this is if the "*" appears inside a character class (see below), in which case it represents the asterisk character itself. Matching zero or more occurrences of some pattern is called a closure. An expression used in a closure will never match NEWLINE.

a*b

Match b, ab, aab, etc.

%a?*b

Match any string that begins with a and ends with b, and that is also the first string in the line. Any number of other characters can come between a and b.

- [A-Z][A-Z]* Match any uppercase word; i.e., any string containing at least two (and possibly more) uppercase characters (see character classes, below). Strings like "Mary" would not match, since "Mary" does not begin with two uppercase characters.
- 6. A string of characters enclosed in square brackets "[string]" is called a character class. This pattern matches any one character in the string but no others. If, however, the first character of the string is a tilde (~), the regular expression matches any one character except the characters in the string. If ~ is not the first character in the string, then it simply matches the tilde character. Note also that the other special characters % \$? * lose their special meaning inside square brackets, and simply represent themselves.

[sam] Match the single characters s, a, or m. (If you want to match the word "sam", don't use the square brackets.)

[~sam] Match any single character except s, a, or m.

7. Within a character class, you can specify any one of a range of letters or digits by indicating the beginning and ending characters in the range separated by a hyphen. That is, 0 through 9 matches any single digit; a through z or A through Z matches any single letter, lowercase or uppercase respectively. (Remember, though, that the actual matching search ignores case unless you have used the DM command SC to enable case sensitivity.) The range can be a subset of the digits or letters (i.e., a through n or 3 through 8). The first and last characters of the range, however, must be of the same type: digit, lowercase letter, or uppercase letter. "[A-9]" is illegal.

Note that the "-" character has a special meaning inside square brackets. If you want to include the literal hyphen character in the class for matching, it must either be the first or last character in the class (so that it does not appear to separate two range-marking characters) or be escaped (see below).

The "]" character is also special to character classes -- it closes the class descriptor list. If you want to include the right bracket character in the class, it also must be escaped (see below).

In summary, the following characters have special meaning inside square brackets: \sim -]

[a-d] Match any single occurrence of a, b, c, or d.

%[A-Z] Match any capital letter that is also the first character on the line (%).

1-[1-9][0-9]* Match any of the page numbers in this chapter.

[0A-Z] Match any string containing a zero or a capital letter.

[~a-z0-9] Match any uppercase letter or punctuation mark (i.e., no lowercase letter or number).

8. The at sign (@) is an escape character. Characters preceded by the "@" character have special meaning in regular expressions, as indicated below.

@n Match NEWLINE character.

@t Match a tab character. Note, however, that the keyboard TAB key does not insert a literal tab; rather it moves the cursor to the display's next tab position. In a regular expression, @t matches only tab characters that have been inserted with @t.

@f Match a form feed character.

In addition, the escape character may be used inside a character class definition ([]) to specify literal occurrences of characters like "-" and "]", which have special functions inside square brackets. You may also use it whenever you need a literal occurrence of some special character in a normal expression (like?, *, or @ itself).

[A-Z@-@]] Match any capital letter, a hyphen, or a right bracket.

@?@* Match a question mark followed by an asterisk, rather than zero or more occurrences of any character (?*).

- 9. You can concatenate regular expressions to form a more complex regular expression. The resulting regular expression matches the concatenation of the strings that the component regular expressions match. All of the examples above concatenate expressions (single characters of some sort) into longer strings for matching.
- 10. You can "tag" parts of a regular expression to help rearrange pieces of a matched string. A text pattern surrounded by braces "{pattern}" is remembered and can be referred to by "@n", where n is a single digit referring to the string remembered by the nth pair of braces.

s/{???}{?*}/@2@1/

S is the DM command for string substitution. The example will move a three-character sequence from the beginning of a line to the end of the line. "???" matches the first three characters of the line, and "?*" matches the rest of the line.

so/{?}{?}/@2@1/ SO is also a DM command for string substitution, but it only substitutes the first occurrence of the first pattern on a line. The example will transpose two characters beginning with the one under the cursor. This can be a handy key definition if you often type "ie" for "ei", etc.

Summary of Features

```
Literal character
%
         Beginning of line (if first character only)
$
         End of line (if last character only)
         Any single character except NEWLINE
         Closure (zero or more occurrences of previous
         pattern)
[...]
         Character class (any one of these characters)
        Negated character class (all characters except
         those in brackets or NEWLINE)
[c1-c2]
        Any one of a range of characters from c1 through c2 (must be same type)
         Escaped character (e.g., @%, @[, @*, etc.)
00
{expr}
         Tagged expression for use later in command line
```

CAUTION: Remember that the special characters described above apply only to regular expression operations. Some of these characters also have meanings (often radically different) in Shell commands and other software products. If you are using a regular expression as a part of one of those Shell commands or products, be sure to enclose the expression in quotation marks so that it will not be misinterpreted.

1.5. Key Naming Conventions

Every key on your keyboard (and mouse) has a name; in fact, almost every key has a set of three or four names. One set is the normal one, and is invoked when the key is pressed down. The second set is invoked when the key is released; these names are the up-transition names. The third set is invoked when the key is pressed simultaneously with the SHIFT key; these are the shifted names. Finally, many keys have special functions when they are pressed simultaneously with the CTRL key; these are the control shifted names.

1.5.1. Standard Key Names

The definable keys (see Figures 1-1 and 1-2) have the following names:

Letters and numbers

These are named by their own single character. The capital letters are distinct from the lowercase letters: merely refer to A instead of worrying about "a shifted". When you refer to these keys in a key definition, enter them within single quotation marks.

ASCII Control

These are the standard intraline and interline control keys.

CR : Carriage Return
BS : Back Space
TAB : Tab
TABS : Shifted Tab
^TAB : Control Shifted Tab
ESC : Escape (low-profile only).
Same as '^[' (hex 1B).
DEL : Delete (low-profile only).
Same as '^|' (hex 7F).

Alphabetic Control

These are named ^x, where x is some other valid key name (i.e., ^Y for CTRL/Y and ^N for CTRL/N). There are also six non-alphabetic control characters. Their names must appear in single quotation marks. The names and the hexadecimal values of the keys are: '^[' (hex 1B), '^\' (hex 1C), '^]' (hex 1D), '^~' (hex 1E), '^?' (hex 1F), and '^|' (hex 7F).

DM Function

These keys perform special Display Manager functions. Those on the left side of the keyboard are named L1 through L9 and LA through LF. (Note that the low-profile keyboard has an extra row of keys below L1 through L3. These keys are named L1A, L2A, and L3A.) Their up-transition names are L1U through L9U and LAU though LFU. Their shifted names are L1S through L9S and LAS through LFS. The DM Function keys on the right side of the keyboard are named R1 through R6. Their up-transition and shifted names are formed in the same way that the left-side keys are.

NOTE: Due to internal hardware restrictions, the 880 keyboard does not execute shifted definitions for these keys. The DM will not object if you define shifted operations for them, but nothing will happen when you try to use the shifted definitions.

Program Function

These keys are specially reserved for user program control. They appear at the top of the keyboard and are named F1 through F8, as labeled. Their up-transition names are F1U through F8U. Their shifted names are F1S through F8S. Their control shifted names are ^F1 through ^F8.

Mouse

These are the keys located on the optional mouse pointing device. Their names are M1, M2, and M3. Their up-transition names are M1U, M2U, and M3U. There are no shifted or control shifted names.

Names containing special characters and all ordinary graphic characters must be in single quotes. For example, to define the lowercase x key so that it acts just like the uppercase X key, you would use the following command line:

kd 'x' es 'X' ke

Consequently, however, you may find it difficult to redefine the x key, because the Display Manager enters an X when you press the x key. This is true for all the alphanumeric and special character keys. Although it is possible to change the definitions of these keys, that capability is intended mainly for use in programs. When a program defines a key, the definition applies only while the program is running and only in pads the program controls.

1.5.2. Controlling Keys from Within a Program

Because of the great flexibility provided by our displays and keyboards, many applications programs assume control of these and redefine various capabilities. For example, the OLD_EDFONT character font editor displays several different menus. You select from the

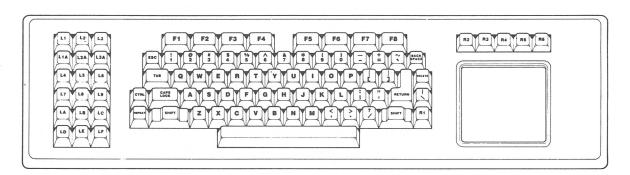


Figure 1-1. The Low-Profile Keyboard Map

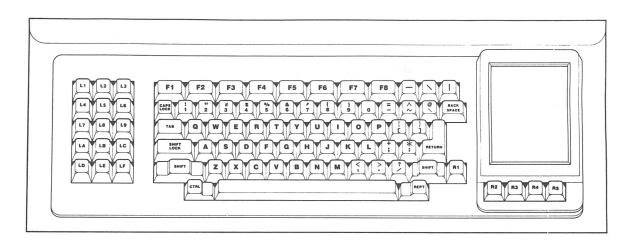


Figure 1-2. The 880 Keyboard Map

menu by pressing one of the Program Function keys (F1 through F8). These keys do not carry their normal DM definitions while OLD_EDFONT is controlling them. Only when your OLD_EDFONT session ends are those keys restored to their previous state.

For your own applications, you may control key definitions through program calls to the PAD_\$DEF_PFK and PAD_\$DM_CMD routines as described in the DOMAIN System Call Reference.

Because the normal functions of the Display Manager keys are often useful (even when they have been redefined by applications programs), the <HOLD> and <HOLD/GO> keys have been defined to provide a temporary override function. Pressing <HOLD> while in an applications program will restore the keyboard to its log in DM definitions. Pressing <HOLD> again will re-enable the application-defined keys.

You may not change this feature of the <HOLD> and <HOLD/GO> keys, which is functional only when the keyboard is under applications program control. This capability is independent of the default Display Manager definitions of WH (WINDOW_HOLD).

1.6. Special Characters in DM Scripts and Key Definitions

Several rules governing the use of literal and special characters affect the proper interpretation of commands within the Display Manager environment. The following characters have special meanings when they appear in a DM command line or script.

The escape character "@" always nullifies any special meaning that the following character might have. As a part of command parsing, the Display Manager strips off the "@" character itself. If you can't remember whether a character has some special meaning to the Display Manager, it is always safe to escape the character -- if it is not special, the Display Manager still removes the "@", so the character appears as it should. The need for character escaping is generally confined to search and substitute operations, commands requiring

quoted stings, and key definitions.

#

&

The use of "@" can be confusing in key definitions because the text in key definitions is actually processed twice - once when the definition is made, and once when the key is depressed and the definition is used. If a character needs to be escaped both times, it must be preceded by three "@" signs. For example, "@@@#" becomes "@#" in the key definition, which then becomes "#" when the definition is used. Only the characters listed in this section are special within key definitions.

When read from a DM script, (via the CMDF comamnd), the "#" character causes the remainder of the line to be treated as a comment and skipped.

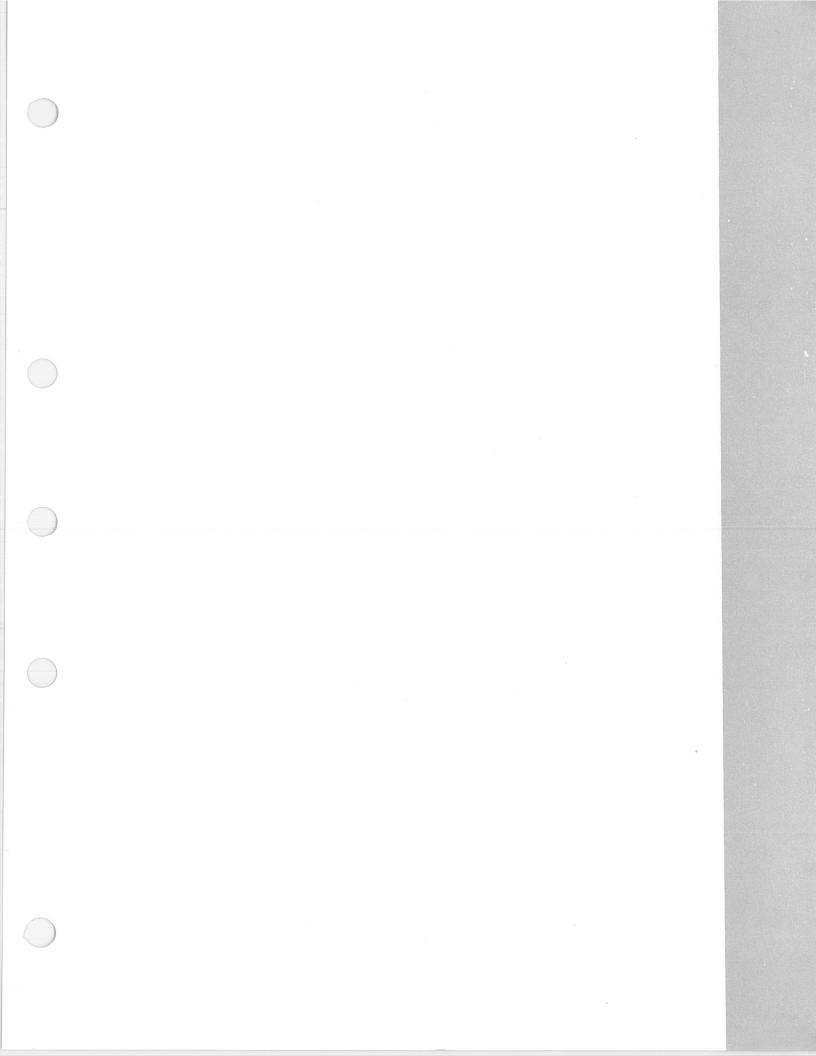
Semicolon is the normal command delimiter. It is equivalent to NEWLINE (generated by <RETURN>).

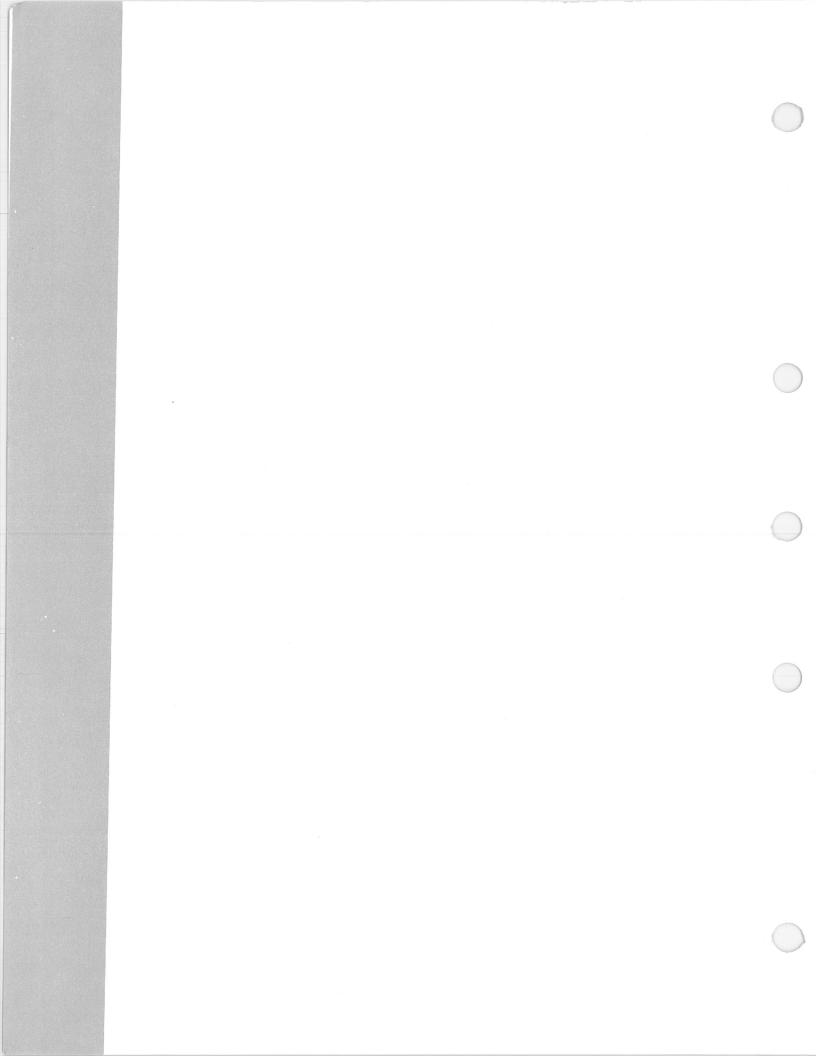
This character makes an input request, except when it is read from the keyboard. When read from the keyboard, it can be used in the replacement part of a substitution command to represent the entire string matching the regular expression. When "&" is preceded by "@" it becomes an ordinary character in both contexts. Therefore, you can't use "&" within a script or key definition and also use its special meaning within substitute commands that appear in that script or definition.

Some commands accept strings surrounded by single quotes. They are KD, ES, CP, CPO, CPS, and the "&" character. When you use single quotes, the only characters in the quoted string that retain their special meanings are "@", "&", and the closing single quote. All other characters revert to their literal graphic values. Note, however, that the KD command is not aware of single quotes within the definition string, so "#" and ";" must be quoted there as well.

For example, to define the F4 key to enter the string "-#-" at the current cursor position, place the following line in a key definition file:

kd F4 es '-@@@#-' ke





TASK LIST REFERENCE SET

MOVING THE CURSOR

Task	DM Command		Predefine	ed Key	
*		Low-Profile		880 Keyboard	
Move left one character	AL	←	(LA)	←	(LA)
Move right one character	AR	\rightarrow	(LC)	\rightarrow	(LC)
Move up one line	AU	1	(L8)	†	(L8)
Move down one line	AD	Ţ	(LE)	Ţ	(LE)
Set arrow key scale factors (raster units)	AS x y	none		none	
Move to start of next line	AD;TL	CTRL/K		CTRL/K	
Move to beginning of line	TL	←	(L7)	←	(L4)
Move to end of line	TR	\rightarrow	(L9)	\rightarrow	(L6)
Move to top line in window	TT	<shift>/</shift>	(LDS)	none	
Move to bottom line in window	TB	<shift>/↓</shift>	(LFS)	none	
Move to window border	TWB {-L, -R, -T, -B	} none		none	
Tab left	THL	CTRL/ <tab></tab>		CTRL/ <tab></tab>	
Tab right	TH	<tab></tab>		<tab></tab>	
Set tabs	TS [n1 n2] [-R]	none		none	
Move to DM Input Pad	TDM	<cmd></cmd>	(L5)	<cmd></cmd>	(L5)
Move to next window on screen	TN	<next wndw=""></next>	(LB)	<next wndw=""></next>	(LB)
Move to previous window	TLW	CTRL/L		CTRL/L	
Move to next window in which input is enabled	TI	none		none	
Move cursor to next unobscured icon on the display	TNI	none		none	

CREATING A PROCESS

Task	DM Command	Predefi Low-Profile	ned Key 880 Keyboard
Create new process, transcript pad, and windows	CP pathname	<shell> (L5S)</shell>	<shell> (R2)</shell>
Create new process without transcript pad or windows	CPO pathname	none	none
Create server process (valid only in DM startup file)	CPS pathname	none	none

CONTROLLING A PROCESS

Task	DM Command	Predefined Key		
		Low-Profile	880 Keyboard	
Quit, stop, or blast a process	DQ [-S -B -C nn]	CTRL/Q	CTRL/Q	
Suspend execution of a process	DS	none	none	
Resume execution of suspended process	DC	none	none	

CREATING A WINDOW READ AND EDIT PADS

Task	DM Command		Predefine	d Key	
	Low-Profile 880 Key				
Create edit pad and window in which to view it	CE pathname	<edit></edit>	(R4)	<edit></edit>	(R4)
Create view window (read-only pad)	CV pathname	<read></read>	(R3)	<read></read>	(R3)
Create copy of existing window	cc	none		none	

MANAGING WINDOWS

Task	DM Command		Predefine	d Key	
		KBD 2 (map)	KBD 1 (map)
Grow or shrink a window using rubberbanding	WGE	<grow></grow>	(L3A)	CTRL/G	
Move a window using rubberbanding	WME	<move></move>	(L3AS)	CTRL/W	
Push or pop a window on stack	WP	<pop></pop>	(R1)	CTRL/P	
Close (delete) a window	WC [-Q -F]	none		none	
Close window, pad; update file	PW;WC -Q	<exit></exit>	(R5)	CTRL/Y	
Close window, pad; no update	WC -Q	<abort></abort>	(R5S)	CTRL/N	
Set scroll mode	WS [-ON -OFF]	CTRL/S		CTRL/S	
Set autohold mode	WA [-ON -OFF]	none		none	
Toggle scroll and autohold modes	WA; WS	CTRL/A		CTRL/A	
Set hold mode Set insert mode (see "Editing a Pad")	WH [-ON -OFF]	<hold></hold>	(R6)	<hold go=""></hold>	(R5)
Define position of default window n	WDF [n]	none		none	
View latest output in Shell's process transcript pad	AU; AU; PB; TI	CTRL/V		CTRL/V	
Copy text to Shell's process input pad	DR;TR;XC;TL;TI;TB; TR;XP;TR	<again></again>	(R2)	<f8></f8>	
Copy text to DM input pad for	DR;	<shift>/</shift>	<read></read>	none	
use with <read></read>	/[~a-z0-9\$@/@~`]/ XC CV FILE;TDM;		(R3S)		
	ES 'CV ';XP CV_FILE; TR;EN				

MOVING A PAD UNDER A WINDOW

Task	DM Command	Predefined Key		
		KBD 2 (map)	KBD 1 (map)	
Move top of pad into window	PT	none	none	
Move cursor to first character in pad	PT;TT;TL	CTRL/T	CTRL/T	
Move bottom of pad into window	PB	none	none	
Move cursor to last character in pad	PB;TB;TR	CTRL/B	CTRL/B	

Save	pad i	n pathname	PN pathname	none	Э	none	
Move	pad n	characters	PH [-]n	<shift>/← (LAS)</shift>	<shift>/→ (LCS)</shift>	← → (L7, L9)	
Move	pad n	lines	PV [-]n	<shift>/↑ (L8S)</shift>	<shift>/↓ (LES)</shift>	F2, F3	
Move	pad n	pages	PP [-]n	↑ (LD,	↓ LF)	↓ ↑ (LD, LF)	

EDITING A PAD

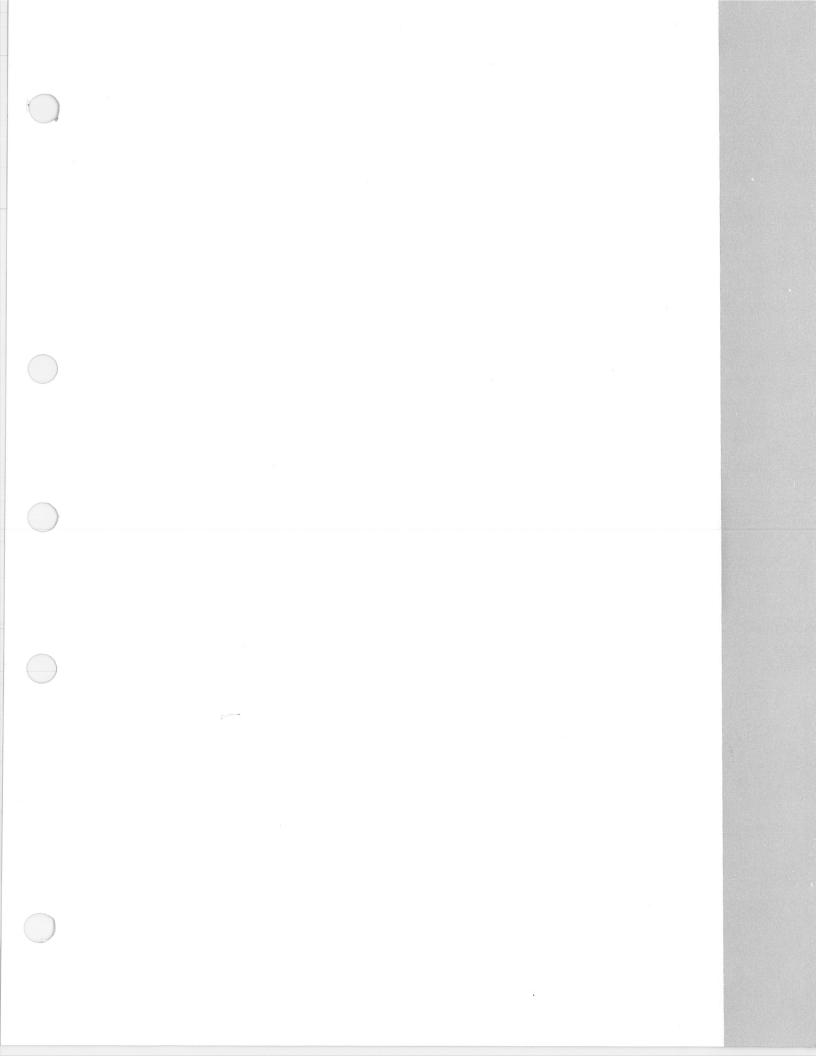
Task	DM Command	Predefined KBD 2 (map)	d Key KBD 1 (map)
Set read/write mode Set insert/overstrike mode	RO [-ON -OFF] EI [-ON -OFF]	CTRL/M <ins> (L1S)</ins>	CTRL/M <ins mode=""> (L1)</ins>
Insert string Insert NEWLINE character Insert new line after current line Insert raw (noecho) character Insert end-of-file mark	ES 'string' EN TR;EN;TL ER nn EEF	[Default DM control of the control o	operation] <return> <f1> none CTRL/Z</f1></return>
Delete character at cursor Delete character before cursor Delete word of text Delete from cursor to end of line Delete entire line .ne 25	ED EE DR;/[-A-ZO-9\$_]/XD ES ' ';EE;DR;TR;XD;TL;TR CMS;TL;XD	<back space=""> <f6> <f7></f7></f6></back>	<char del=""> (L3) <back space=""> <f6> <f7> <line del=""> (L2)</line></f7></f6></back></char>
Copy text to paste buffer Cut (delete) text and write it to paste buffer Paste (write) text in paste buffer into pad Copy a portion of the screen into a gmf file	<pre>XC [-F pathname name] XD [-F pathname name] XP [-F pathname name] XI [pathname]</pre>	<copy> (L1A) <cut> (L1AS) <paste> (L2A) none</paste></cut></copy>	CTRL/C CTRL/E CTRL/O none
Search forward for string Repeat last forward search Search backward for string Repeat last backward search Abort search Set case comparison for search	/string/ // \string\ \\ ABRT (or) SQ SC [-ON -OFF]	none CTRL/R none CTRL/U CTRL/X none	none CTRL/R none CTRL/U CTRL/X none
Substitute string2 for all occurrences of string1 in defined range Substitute string2 for first occurrence of string1 in each line of defined range	S/string1/string2/ SO/string1/string2/	none	none
Undo previous command(s)	UNDO	<undo> (L2AS)</undo>	none
Update edit file without closing edit pad	PW	<save> (R4S)</save>	none

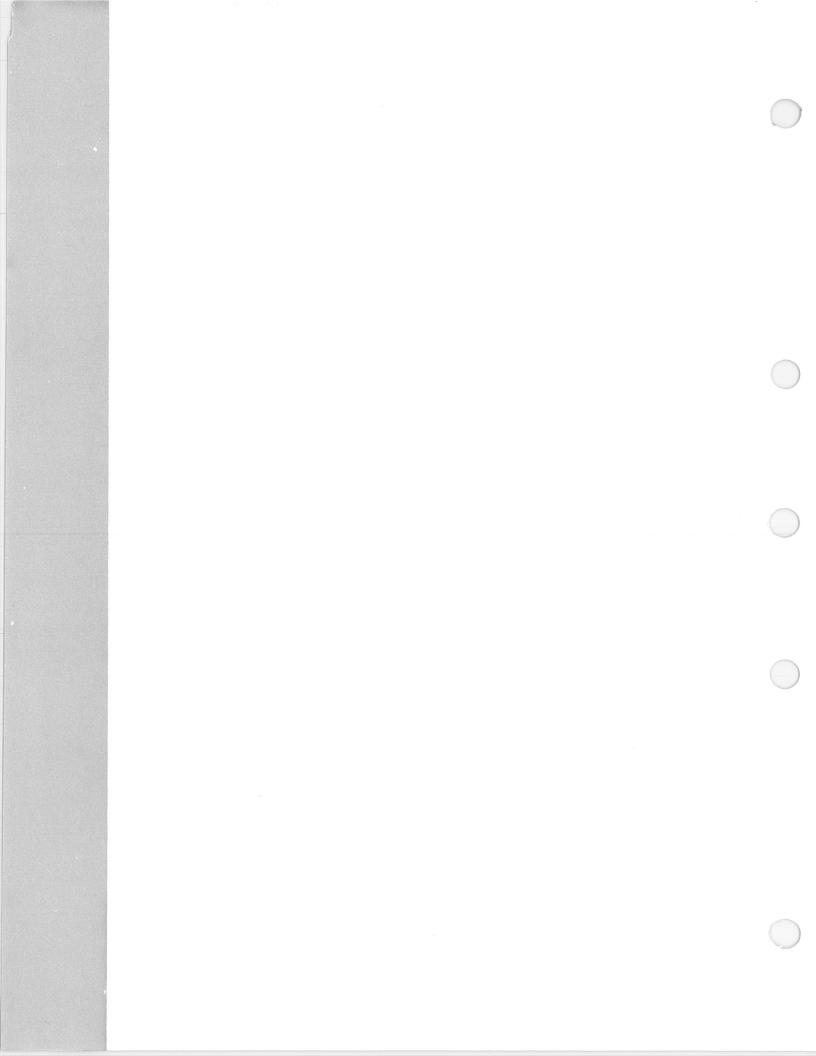
USING WINDOW GROUPS AND WINDOW ICONS

Task	DM Command	Predefined Key	
		Low-Profile	880 Keyboard
Add a window or group to a new or existing group of windows	WGRA group_name [entry_name]	none	none
Remove a window or group from a group of windows	WGRR group_name [entry_name]	none	none
Make a window visible or invisible	WI [entry_name][-I -W]	none	none
Change a window into an icon or an icon into a window	<pre>ICON [entry_name][-I -W]</pre>	none	none
Set icon position and offset vector	IDF	none	none
Display a list of the windows in a group	СРВ	none	none

MANAGING THE DISPLAY ENVIRONMENT

Task	DM Command	Predefin KBD 2 (map)	ed Key KBD 1 (map)
Request help	none	<help> (R6S)</help>	none
Log in Log off Shut down system	L id [proj [org]] [-P] LO [-F] [-ON -OFF] SHUT [-F]	[-H] none none none	none none none
Place a mark Go to a mark Clear mark stack	DR GM CMS	<mark> (L1) none none</mark>	<mark> (R1) none none</mark>
Display cursor coordinates (line, column; x/y coord.)	=	none	none
Acknowledge DM alarm Acknowledge alarm and pop window	AA AP	none none	none
Set or display an environment variable	ENV var [value]	none	none
Set background color Set window color	BGC [-ON -OFF] INV [-ON -OFF]	none none	none none
Refresh entire screen Refresh window	RS RW [-R]	CTRL/F none	CTRL/F none
Load font for use in pads	FL pathname	none	none
Declare keyboard type Set or display key definition Request input to command line	KBD [type] KD name [[def.] KE] &'prompt'	none none none	none none none
Execute DM script	CMDF pathname	none	none





Chapter 2 Display Manager Commands

AA (ACKNOWLEDGE_ALARM)

AA (ACKNOWLEDGE_ALARM) -- Acknowledge Display Manager Alarms

FORMAT

$\mathbf{A}\mathbf{A}$

The AA command acknowledges a Display Manager alarm. This command turns off the current alarm and enables further alarms, which may already be waiting. AA requires no arguments or options.

ABRT (ABORT) -- Abort text search; cancel any action involving ECHO

FORMAT

ABRT (CTRL/X)

The ABRT command aborts a text search, and cancels any action involving the ECHO command.

When you use ABRT to abort the current search, the Display Manager returns the message "Search aborted." It does not move the window. Note that you cannot type this command during a search.

When you use ABRT to abort the ECHO command, ABRT cancels a move window with rubberbanding or grow window with rubberbanding operation; or, it cancels highlighting for a defined range of text, depending on how ECHO was used.

CTRL/X issues this command by default.

AD (ARROW_DOWN)

AD (ARROW_DOWN) -- Move cursor down one line.

FORMAT

AD

The AD command moves the cursor down one line from its current position. By default, the down arrow key (LE) on the left hand key pad executes this command.

AL (ARROW_LEFT) -- Move cursor left one character.

FORMAT

AL

The AL command moves the cursor left one character from its current position. By default, the left arrow key (LA) on the left hand key pad executes this command.

AP (ACKNOWLEDGE_POP)

AP (ACKNOWLEDGE_POP) -- Acknowledge alarm and pop window.

FORMAT

AP

The AP command acknowledges a Display Manager alarm and displays ("pops") the window to which the alarm pertains. This command is particularly useful if the window is completely covered so that you cannot point to it. AP requires no arguments or options.

AR (ARROW_RIGHT) -- Move cursor right one character.

FORMAT

AR

The AR command moves the cursor right one character from its current position. By default, the right arrow key (LC) on the left hand key pad executes this command.

AS (ARROW SCALE)

AS (ARROW_SCALE) -- Set scale factors for arrow keys.

FORMAT

AS [x[y]]

The AS command sets scale factors for the arrow keys. The scale factor is useful for changing the apparent sensitivity of the arrow keys and for lining up the edges of windows after moving them.

ARGUMENTS

If no arguments are specified, then the default scale factors are used, as described below.

x (optional)

Specify horizontal scale factor in raster units (integer). This value must be in the range 0-1023. (Note, however, that portrait displays may only display up to 800 raster units in this dimension.) There are approximately 100 raster units per inch. The default horizontal movement is the width of the character on which the cursor rests; if the cursor is not on a character, the DM uses the width of a space in the last window. Specifying 0 for 'x' indicates that the default should be used.

Default if omitted: 0

y (optional)

Specify vertical scale factor in raster units (integer). This value must be in the range 0-1023. (Note, however, that landscape displays may only display up to 800 raster units in this dimension.) The default vertical movement is the height of a line in the last window. Specifying 0 for 'y' indicates that the default should be used.

Default if omitted: leave current y value unchanged

AU (ARROW_UP) -- Move cursor up one line.

FORMAT

AU

The AU command moves the cursor up one line from its current position. By default, the up arrow key (L8) on the left hand key pad executes this command.

BGC (BACKGROUND_COLOR)

BGC (BACKGROUND_COLOR) -- Set background color of display.

FORMAT

BGC [-ON | -OFF]

The BGC command sets the display's background color to grey or green (-ON) or black (-OFF). Entering BGC without an option toggles the current mode. The background color is ON, by default, at login.

NOTE

BGC has no effect on nodes with color displays.

CASE -- Change case of all letters in a defined range of text.

FORMAT

[range]CASE [options]

The CASE command changes the case of all the letters in a defined range of text. You can instruct CASE to invert the case of all letters, change all letters to uppercase, or change all letters to lower case. If you do not specify a range, CASE operates on the text from the cursor position to the end of the current line.

OPTIONS

-S	Swap all uppercase letters for lowercase, and all lowercase letters for upper case (in the defined range).
- U	Change all letters in the defined range to uppercase.
-L	Change all letters in the defined range to lowercase.

CC (CREATE_COPY)

CC (CREATE_COPY) -- Create a copy of an existing window.

FORMAT

CC

The CC command creates a copy of an existing window. To use this command, mark both edges of the new window with the DR command, then point to the window to be copied. With the cursor in the window to be copied, press < CMD> and issue the CC command.

If no region is marked, the new window is created using the next default DM window.

NOTE

There is a homonymous Shell command: CC (COMPILE_C) -- Compile a C program. See the CC command description in the DOMAIN C Language Reference for more information.

CE (CREATE EDIT) -- Create an edit pad and window

FORMAT

[region]CE pathname [options]

Giving the CE command causes the Display Manager to create an edit pad and a window in which to view it. If the file specified exists, it is opened for editing. If the file does not exist, the DM creates and opens a file with the specified name.

By default, the <EDIT> key (R4) invokes the CE command, automatically moving the cursor to the DM input pad and issuing the "Edit file: " prompt. Type the pathname of the file to be edited.

Once an edit pad is created, you may use other DM commands to manipulate text in the pad. See the DM index pages at the end of the previous chapter for a list of editing commands.

To close a pad and window, use the DM command WC (usually CTRL/N).

NOTE: CE does not create a process. It simply opens a file for editing within the current Display Manager process.

ARGUMENTS

		۰		
re	g	1	0	n

(optional)

Specify area of the screen where the new window will be displayed. For details on window boundaries, see the previous chapter.

Default if omitted: use next DM default window.

pathname

(required)

Specify file to be edited.

OPTIONS

-I

Specify that the window created for this pad will be in icon format.

-C 'char'

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

CMDF (COMMAND_FILE)

CMDF (COMMAND_FILE) -- Execute DM script

FORMAT

CMDF pathname

CMDF directs the Display Manager to read commands from a file (DM script). When it reaches the end of the file, the cursor returns to its previous location.

Command files may be nested; i.e., CMDF may be used within another DM script.

ARGUMENTS

pathname

(required)

Specify name of file to be executed. DM commands may appear one per line, or several per line, each delimited by semicolons.

CMS (CLEAR_MARK_STACK) -- Erase existing marks.

FORMAT

CMS

The CMS command erases any existing marks. Use it to ensure that commands requiring marked regions do not behave unexpectedly as a result of outstanding (but probably forgotten) marks. The <LINE DEL> standard key definition is a good example: "CMS;TL;XD". Previous marks are cleared so that only the current line is deleted.

CMS requires no arguments or options.

CP (CREATE PROCESS) -- Create process, pads, and windows.

FORMAT

[region]CP [options] pathname [args ...]

CP creates a process, input and transcript pads, and input and transcript windows, then executes a program (indicated by the pathname argument) within that process. The transcript pad is opened as standard output in the new process. To create a process running the Shell that we supply, either press the <SHELL> key (which causes everything to happen automatically) or specify /COM/SH for pathname as follows:

Command: CP /COM/SH <RETURN>

This command creates an input pad associated with the transcript pad, and opens it as standard input. As a result, you have a new process, windows to its input and transcript pads, and a shell.

To stop a shell and delete all windows and pads associated with its parent process, type CTRL/Z in the shell's process input pad, then CTRL/N (refer to the DQ command in the next section for more information).

ARGUMENTS

re	gı	on

(optional)

Specify area of the screen where the new window will be displayed. For details on window regions, see the previous chapter.

Default if omitted: use next DM default window.

pathname

(required)

Specify file to be executed by the new process: usually a Shell

(command interpreter).

args ...

(optional)

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those

arguments in quotation marks.

OPTIONS

Note that options, if present, must precede the 'pathname' argument.

-I Specify that the window created for this process will be in icon

format.

-C 'char' Specify the icon character to be used in the icon window. 'char'

must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the

default icon character for this pad type.

-N name

Assign process name 'name.' If omitted, the DM assigns the name "Process_n," where n is an integer beginning with 1 and incremented by 1 for each active process. Due to an error in the DM at SR8, the -N option must FOLLOW the 'pathname' argument (and any 'args' arguments) in order to be interpreted correctly.

EXAMPLES

1. Create a process named 'spare' running the Shell. The '-nstart' option on SH suppresses startup file execution for the new Shell.

Command: (0,0)dr; (500,300)cp /com/sh -nstart -n spare

2. Create a process running the Shell, and place it in a window in icon format using the default icon character for this pad type.

Command: cp -i /com/sh

2-17

CPB (CREATE_PASTE_BUFFER) -- Display a list of the windows in a group.

FORMAT

CPB group_name [options]

The CPB command creates a window on a named paste buffer specific to the given group. The paste buffer contains a list of the windows in the group. Because these group lists are held in paste buffers, your programs can access the groups by using the PBUFS routines described in the DOMAIN System Call Reference.

The DM automatically creates three special paste buffers to help you manage your windows and icons. These paste buffers contain the following groups:

- The INVIS_GROUP -- this buffer holds the pathnames of all the windows that you have made invisible.
- The ICON_GROUP -- this buffer holds the pathnames of all the windows that are represented by icons.
- The ALL_GROUP -- this buffer holds the pathnames of every window open on your node - Shell process windows, DM windows, visible and invisible windows, and windows represented by icons.

These special groups are created regardless of any other groups, and their members may overlap with the members of any other group (just as any group can have the same member(s) as another).

A special feature of the CPB command allows you to directly access the windows in a group when the paste buffer holding the group is displayed on your screen. To use this feature:

- 1. Use the CPB command to display the list of windows.
- 2. Position the cursor on the pathname of the window you want to access.
- 3. Press < CMD>, and issue the DR (MARK) command.
- 4. Press < CMD > again, and issue the desired DM command.

By using this feature you can directly access windows that are invisible, represented with icons, etc.

ARGUMENTS

group_name
(required)

Specify the name of the group you want to display.

OPTIONS

-I

Specify that the window created will be in icon format.

-C 'char'

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the default icon character for this pad type.

CPO (CREATE_PROCESS_ONLY) -- Create process without pads or windows.

FORMAT

CPO pathname [options]

The CPO command creates only a process, without associated pads or windows. The four standard I/O streams are directed to /DEV/NULL. If this command appears in the node's DM boot startup script 'NODE_DATA/STARTUP the system assigns the new process the subject identifier (SID) USER.SERVER.NONE.local_node, and the created process will continue to run regardless of whether or not any one is logged in. This is desirable for utilities like the PRSVR (PRINT_SERVER) and NETMAN, and means that CPO is identical to CPS in this context. (This is only effective at version 6.0 and later.)

If CPO is issued in any other startup script or from the keyboard, the SID of the new process is derived from whatever process invokes CPO, and the created process will terminate at logout.

ARGUMENTS

pathname

(required)

Specify file to be executed by the new process.

OPTIONS

-N name

Assign process name 'name'. If omitted, the process is not

named.

args . . .

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those

arguments in quotation marks.

EXAMPLES

1. Run the ALARM SERVER in a background process.

Command: cpo /sys/alarm/alarm_server -disk 98 -bell1

CPS (CREATE_PROCESS_SERVER) -- Create process independent of login.

FORMAT

CPS pathname [options]

CPS creates a process (without associated pads or windows) that runs regardless of whether or not any one is logged in. This is desirable for utilities like the PRSVR (PRINT SERVER) and NETMAN. CPS may appear in any of the DM startup scripts. You may prefer to issue the CPS command from the keyboard on selected occasions, however, rather than include this function in a startup script.

The assigned the subject identifier created process is USER.SERVER.NONE.local node regardless of the context in which the CPS command appears. Be sure that any files to be used by this process (including the program specified by the 'pathname' argument) give adequate access to this SID. If the ACLs on the files do not allow proper access to the SERVER project name, the process will terminate. And since background processes are essentially invisible, no error messages are returned to the display, making fault diagnosis difficult.

ARGUMENTS

pathname

(required)

Specify file to be executed by the new process.

OPTIONS

-N name

Assign process name 'name'. If this option is omitted, the

process is not named.

args . . .

Specify any arguments to be passed to the program 'pathname'. If any of these arguments contain explicit blanks, enclose those arguments in quotation marks.

EXAMPLES

1. Run the server MBX HELPER.

Command: cps /sys/mbx/mbx helper -n mbx helper

CV (CREATE_VIEW) -- Create a read-only edit pad and window.

FORMAT

[region] CV pathname

The CV command creates a read-only edit pad to view an existing file. You may not make changes to the file, only view it. If you decide that you want to make changes to it after all, you must first disable read-only mode. See the RO command description for details about that operation.

By default, the <READ> key (R3) invokes the CV command, automatically moving the cursor to the DM input pad and issuing the "Read file: " prompt. Type the pathname of the file to be read.

To close a pad and window, use the DM command WC (usually CTRL/N).

ARGUMENTS

PA	OTI	0	m
	page A	v	4.

(optional)

Specify area of the screen where the new window will be displayed. For details on window regions, see the previous

chapter.

Default if omitted: use next DM default window.

pathname

(required)

Specify file to be viewed. An error occurs if the file does not

exist.

OPTIONS

-I

Specify that the window created for this pad will be in icon format.

-C 'char'

Specify the icon character to be used in the icon window. 'char' must reside in the current icon font. If this option is not specified and -I is present, the Display Manager will use the

default icon character for this pad type.

DC (DEBUG_CONTINUE) -- Continue a suspended process.

FORMAT

DC

The DC command restarts a process that has been suspended by the DS (DEBUG_SUSPEND) command. Refer to the DS command description for details about that operation.

DC requires no arguments or options.

DQ (DEBUG_QUIT) -- Generate a quit fault in a process.

FORMAT

DQ [options]

The DQ command generates a quit fault, which normally interrupts execution of the current program and returns the process to the calling program (usually the Shell). This command affects the process associated with the window that contains the cursor. By default, CTRL/Q invokes DQ without options to generate a normal quit fault in the program currently running.

OPTIONS

If no options are specified, then DQ generates a normal quit fault and halts whatever program is currently running.

-C nn	Generate an arbitrary asynchronous fault with the specified hexadecimal status (nn).
-S	Stop entire process in a controlled way, if possible. Close open streams, files, pads, etc. The Shell's parent process is stopped and closed, too.
-B	Blast process; do not execute further user mode instructions.

NOTE: If you are trying to stop a Shell's parent process (as opposed to some program running within a Shell), there is an easier method than typing DQ -S. Position the cursor in the Shell's process input window and issue an EEF (END_OF_FILE) command (by default, CTRL/Z). This signals completion of input, and stops both the Shell and the process.

DR (DEFINE_REGION) -- Place a mark to define a region.

FORMAT

DR

The DR command marks some part of the display or some part of a pad. The mark can be used to define a region for a substitute command, to grow, shrink, or move a window, or to reposition the cursor.

You may specify a literal point at which the mark is to be placed by preceding the DR command with line and column numbers in a pad, x and y screen coordinates, or regular expressions for matching text. If no point is specified, the mark is placed at the current cursor position.

By default, the <MARK> key invokes the DR command along with ECHO to provide user-visible feedback.

DS (DEBUG_SUSPEND)

DS (DEBUG_SUSPEND) -- Suspend a process.

FORMAT

DS

The DS command generates a temporary interrupt for a process. All activities are suspended. Processes may be restarted with the DC (DEBUG_CONTINUE) command. See the DC command description for details.

DS requires no arguments or options.

ECHO -- Begin text echoing, end rubberbanding.

FORMAT

ECHO [option]

When used as part of the DR; ECHO command sequence invoked with <MARK>, the ECHO command performs two seperate operations depending on the situation where it is used. When you press <MARK> to begin defining a range of text, ECHO tells the DM to begin highlighting the indicated text range in reverse video (text echoing). When you use <MARK> to complete a move window (WME) or grow window (WGE) operation, the ECHO command tells the DM to remove the "rubberband" and move or grow the window as indicated. You can abort text highlighting or rubberbanding using the SQ command. (By default, the CTRL/X key combination issues the SQ command.)

OPTIONS

-R

Specify ECHO for a rectangular region of text. Use a mark point and the cursor to specify a column along the left side of the text you want to highlight in reverse video. When you issue the ECHO command with the -R option all text to the right of the specified column is then diplayed in reverse video.

ED (EDIT_DELETE)

ED (EDIT_DELETE) -- Delete character under cursor.

FORMAT

ED

The ED command deletes the character under the cursor. If the character is a NEWLINE, ED joins two lines. By default, the <CHAR DEL> key invokes the ED command.

ED requires no arguments or options.

NOTE: There is a homonymous Shell command: ED -- Invoke line mode editor. See the ED command description in the Shell chapter for details.

EE (EDIT_ERASE) -- Delete character preceding cursor.

FORMAT

EE

The EE command deletes the character preceding the cursor. If the window is in overstrike mode, EE replaces the preceding character with a blank. By default, the <BACKSPACE> key invokes the EE command.

EE requires no arguments or options.

EEF (EDIT_END_OF_FILE)

EEF (EDIT_END_OF_FILE) -- Insert end-of-file mark.

FORMAT

EEF

The EEF command inserts a stream end-of-file mark (EOF) in the pad. If the line containing the cursor is empty, the end-of-file mark is written on that line. Otherwise, the end-of-file mark is inserted following the current line.

By default, CTRL/Z executes the EEF command.

It is a common (although not universal) convention for programs to terminate execution and return to the process which called them when they receive an EOF on their standard input stream. The command Shell is such a program. When the top-level program in a process (usually /COM/SH) returns, the process stops and all of its streams are closed. The Display Manager then closes the Shell's process input pad and window, and closes the transcript pad. Whether or not the transcript window also disappears depends on the setting of its auto-close mode (see the WC command description for details). If auto-close is disabled (the default condition), then you must manually delete any windows associated with the closed transcript pad by using the DM command WC -Q, or CTRL/N.

EI (EDIT_INSERT) -- Set insert/overstrike mode.

FORMAT

EI [-ON | -OFF]

The EI command puts the current pad into (-ON) or out of (-OFF) insert mode. If no option is supplied, the current mode is inverted. In insert mode, characters you type are inserted into the pad without replacing or overstriking any existing characters. This causes existing text to drift to the right as new text is added. In overstrike mode (i.e., insert mode turned off), characters typed at the keyboard replace those under the cursor. This can be useful for entering information into pre-formatted files so that the format is undisturbed.

By default, the <INS> key on low-profile keyboards and the <INS MODE> key on 880 keyboards invoke the EI command without options to toggle the current mode.

The window legend contains an "I" when the window is in insert mode. The "I" disappears in overstrike mode.

All pads are initially in insert mode, although this is irrelevant if the pad is also read-only.

EN (EDIT_NEWLINE)

EN (EDIT_NEWLINE) -- Insert NEWLINE.

FORMAT

EN

The EN command inserts (or overstrikes, depending on current mode) a NEWLINE character at the current cursor position.

By default, the <RETURN> key invokes this command.

ENV (ENVIRONMENT) -- Set or display an environment variable.

FORMAT

ENV variable [value]

The DM command ENV sets or displays the value of an environment variable. Environment variables are of primary concern to DOMAIN/IX users; please consult the DOMAIN/IX documentation for details about their usage.

If you invoke ENV from the keyboard, you may use it only to display environment variables, not set them. To set variables, ENV must appear in one of your startup scripts so that it gets executed before any Shells are created. This is because the DM assigns values to environment variables for new Shells using those in effect for the window that currently contains the cursor. ENV thus does not have a chance to influence the new Shell if other Shell(s) already exist. In addition, the ENV command will NEVER change the value of a variable in an existing process.

For details about manipulating environment variables from the Shell, see the EXPORT command description in the Shell commands chapter.

ARGUMENTS

variable

(required)

Specify the name of the variable whose value is to be set or displayed. Since the DM normally forces arguments to uppercase prior to command scanning, enclose a variable whose name must be lowercase in single quotation marks.

value

(optional)

Specify the new value to be assigned to 'variable'. Since the DM normally forces arguments to uppercase prior to command scanning, enclose a value which must be lowercase in single quotation marks.

Default if omitted: display the current value of 'variable'.

EXAMPLES

Command: env SYSTYPE

Display the current value for SYSTYPE for the current Shell window.

Command: env SYSTYPE 'bsd4.2'

Set the SYSTYPE variable to 'bsd4.2'. This line must appear in a startup script to have any effect.

ER (EDIT_RAW)

ER (EDIT_RAW) -- Insert raw character.

FORMAT

ER nn

The ER command sends a raw character to a program. The single argument nn (required) is a one or two character hexadecimal value which defines the single byte sent to the active program the next time the program requests input. The data byte is not echoed anywhere on the display. In effect, this command delivers a single raw keystroke to a program.

This command can be used by programs that need to define keys to return known values for actions by the programs.

This command differs from the other text insertion commands in that it does NOT insert the hexadecimal character into an edit pad. Its sole function is to pass a hexadecimal character to a running program. ES (EDIT_STRING) -- Insert string.

FORMAT

ES 'string'

If a window is currently in write mode, then any text character typed at the keyboard is inserted at the current cursor position. This is the default Display Manager action. Typing text into a read-only window causes an error.

The ES command inserts a string of text at the current cursor position. Enclose the string to be inserted in apostrophes. Since text insertion is the default action anyway, this command is primarily useful in key definition commands where you want some text written out when the key is pressed, or in DM scripts for writing text to the display.

EX (EXIT)

EX (EXIT) -- Exit Display Manager to Boot Shell.

FORMAT

 $\mathbf{E}\mathbf{X}$

The EX command causes the system to stop the Display Manager process and enter the Boot Shell. This puts you in the same place that you would be if you had powered up your node with the NORMAL/SERVICE switch set to SERVICE.

To restart the Display Manager, type

) GO

in the Boot Shell.

This command differs from SHUT, which shuts the node's operating system (AEGIS) down completely and enters the Mnemonic Debugger that resides in the node's boot PROM.

Do not confuse this command with the Shell command EXIT, which exits a Shell script loop. See the EXIT command description for more information.

FL (FONT_LOAD) -- Load a font for use in pads.

FORMAT

FL pathname [-I]

The FL command loads a font for use in subsequent pads. Note that fonts apply to pads, not windows, so any new window opened to an old pad uses the old font.

You can load up to 50 fonts. The DM keeps track of fonts loaded by the FL command or programs. It unloads fonts on a least-recently-used basis.

If you need to unload a font (to edit it with EDFONT for example), issue an FL command for another font, and close all the windows using the font you wish to unload. If a program loaded the font, stop the program (issue an end-of-file -- usually CTRL/Z), and close the window.

ARGUMENTS

pathname

(required)

Specify name of file containing font to be loaded. The given pathname is first looked up directly, using the user working and naming directory rules. If not found, it is looked up in the directory /SYS/DM/FONTS.

OPTIONS

-I

Specify that the font to be loaded is an icon font.

GM (GO_TO_MARK)

GM (GO_TO_MARK) -- Go to a mark.

FORMAT

GM

The GM command repositions the cursor at the most recently marked point after first marking the current cursor position (where you invoked the GM command). This allows you to alternate between two points with repeated invocations of GM. The most common use of this is to "remember" a position in a file and return to it later. GM repositions the window, if necessary, to display the marked pad location.

Pads may contain multiple marks, in which case GM works its way progressively through the mark stack, consuming each mark as it goes. You may also place marks in different pads; GM builds only one mark stack, not one for each pad. Thus you may use this command to jump around between pads.

GM requires no arguments or options.

ICON -- Change a window or window group into an icon(s); change an icon back into a window.

FORMAT

ICON [entry_name] [options]

The ICON command operates as a toggle and changes the specified window or group into an icon(s), or changes an icon back into a window. You can use two methods to change a window into an icon: specify the window name (from the window legend) when you issue the ICON command, or simply position the cursor in the window, press <CMD>, and issue the ICON command. If you want to change a group of windows into icons, you must specify the group name when you issue the ICON command. By default, if you do not specify an entry name (the name of a window or group) ICON uses the pathname of the window where the cursor was last positioned.

To change an icon back into a window, position the cursor on the icon, press < CMD>, and give the ICON command. The window reappears on your display at its former position.

When you change a window into an icon, the DM displays an icon character that describes the type of information the window displayed, such as an edit pad, a graphics file, or a Shell transcript pad. The default icon characters are held in a font file called /SYS/DM/FONTS/ICONS. If you desire, you can examine or change this file using the EDFONT program described in the appendices.

ARGUMENTS

entry_name	
(optional)	Specify the name of the window or group you want to change into icon(s), or change back into a window.
	Default if omitted: Use the pathname of the window where the

cursor was last positioned.

OPTIONS

-I	Force the window or group to appear as an icon(s).
-W	Force the window or group to appear as a window.
-C char	Specify the the DM icon character used to represent a window.

IDF (ICON_DEFAULT) -- Set the icon default positioning and offset.

FORMAT

IDF

The IDF command sets the position of an icon on your screen, determines where subsequent icons will be positioned (the offset), and specifies the icon shift vector to use when icons start to overlap each other. Each time you issue the IDF command you reset the positions where any subsequent icons appear.

By default, icons appear in a horizontal line along the top of portrait displays, and in a vertical line along the right side of landscape displays. The default offset is set at the width of one icon (60 bits) horizontally or vertically, depending on the display. You can use IDF to change this default positioning and offset, to establish the position of an icon created in a script, or to set your personal icon positioning and offset in a DM startup script (STARTUP_DM). Specify the IDF command in one of the following ways:

• Move the cursor to the new default icon position. Issue the IDF command.

This operation sets the first icon position; the offset of the next icon will be 0,0 (pixels) and the shift vector will be 0,0 (pixels). Therefore, all subsequent icons will appear on top of one another at the first icon position.

• Move the cursor to the new default icon position. Use the <MARK> key or issue the DR command to mark the cursor position. Move the cursor to indicate the offset for the next icon. Issue the IDF command.

This operation sets the first icon position and next icon offset. The shift vector is 0,0 (pixels). Therefore, when icons need to use already-occupied positions, the DM places new icons directly on top of pre-existing icons.

• Move the cursor to the new default icon position. Use the <MARK> key or issue the DR command to mark the cursor position. Move the cursor to indicate the offset for the next icon and once again issue a <MARK> or DR command. Then move the cursor again in order to set the shift vector for reused icon positions.

This operation sets the first icon position and the next icon offset. It also establishes a shift vector so that icons will not appear directly on top of one another if the DM needs to place new icons over pre-existing ones.

• Specify the icon position, offset, and shift explicitly in a command line. The format is as follows:

(first_xy_pos)DR; (next_xy_pos)DR; (shift_xy_pos);IDF
For example, the command line:

Command: (800,10) DR; (850,60) DR; (820,10); IDF places the upper-left corner of the first icon at bit position (800,10), sets the

icon offset vector to (50,50) (found by subtracting the 'initial' from the 'next' bit positions), and sets the shift vector to (20,0) (found by subtracting the 'initial' from the 'shift' bit positions). The next icon would, therefore, appear at bit positions (850,60), the next at (900,110), etc. If an icon must be placed on top of the first icon, then it will be positioned at (820,10), the next at (870,60), etc.

The IDF command requires no arguments or options.

INV (INVERT_COLOR)

INV (INVERT_COLOR) -- Set window color.

FORMAT

INV [-ON | -OFF]

The INV command sets the color of all windows on the screen. At log in, INV is OFF, which gives white (or green) characters on a black background. Setting INV to ON gives black characters on a white (or green) background. Entering INV without an option toggles the current mode. Note that these are window backgrounds only; the display background is controlled with BGC (BACKGROUND_COLOR).

NOTE: INV has no effect on nodes with color displays.

KBD (KEYBOARD) -- Declare keyboard type.

FORMAT

KBD id

NOTE: This command is valid only in the DM file 'NODE_DATA/STARTUP; it may not be typed from the keyboard. See the *DOMAIN System User's Guide* for information on startup files.

KBD allows you to specify that you have the low-profile keyboard attached to your node so that the proper set of standard key definitions may be applied. When this command is invoked in the 'NODE_DATA/STARTUP file, it causes the DM to execute the corresponding key definition file (/SYS/DM/STD_KEYS2). If this command is not invoked, then the key definitions in /SYS/DM/STD_KEYS (corresponding to the 880 keyboard) are used.

ARGUMENTS

id

(required)

Specify keyboard ID. This is only necessary if your node is using the low-profile keyboard, in which case specify the ID as "2".

KD (KEY_DEFINITION) -- Set or display key definition.

FORMAT

KD key_name [[definition] KE]

The KD command defines a keyboard key as a sequence of DM commands. It also can display the definition of a key.

ARGUMENTS

key_name

(required)

Specify the name of the key to be defined or displayed. Key names are available from HELP DM KEYS. Enclose normal alphanumeric and punctuation keys in quotation marks.

definition

(optional)

Specify sequence of DM commands that represent desired key function; separate commands with NEWLINEs or semicolons. Definition can be any number of commands, but cannot exceed 256 characters. Definitions may contain other predefined keys (i.e., key definitions may be embedded in one another).

The input request character, '&', which is frequently used in key definitions, must be preceded by an escape character (@) when the KD command appears in a script.

If 'definition' is not specified and KE is present (i.e., definition is null), then the current key definition is deleted and the key reverts to its normal graphic value, if any. If KE is also absent, then the definition of the named key is displayed in the DM message window.

Default if omitted: see above

KE

(optional)

Signal the end of the KD command. This argument is required if 'definition' is present, or if you wish to delete a definition by specifying a null definition.

Default if omitted: display 'key name' definition

EXAMPLES

1. Command: KD L3

Display definition of key L3.

2. Command: KD F6 AU; TR KE

Define F6 key to move the cursor to end of previous line in window.

3. Command: KD ^C KE

Delete current definition of CTRL/C.

You can embed key definitions in key definitions, and thereby define keys that define other keys. The embedded key definition follows the same rules as any other key definition. The KE that ends the embedded definition must be separated from the next command by an "escaped" semicolon; that is, a semicolon preceded by the @ character. For example:

Command: KD F3 KD 'X ES 'April is the cruelest month' KE@; PV KE

changes the definition of the F3 key, which normally just invokes the DM command PV, so that it also changes the definition of CTRL/X to print out the string shown. If the ';' were not preceded by an escape character, the definition would not be accepted.

Note that key definitions within key definitions are scanned THREE times: 1) when the outer key definition is made, 2) when the outer key definition is executed and the inner key definition is made, and 3) when the inner key definition is executed. Because of this, you must exercise great care when escaping (with "@") certain special characters such as "@" itself.

L (LOGIN) -- Log in to a node.

FORMAT

L id [proj [org]] [options]

The L command allows you to log in to a node. It is valid only at the beginning of a session in reponse to the "Please log in:" prompt. Typing the L command after logging in causes an error. After entering the L command, the system will request a password. If you specify either the ID or the password incorrectly, the system displays an error message and the correct format of the L command, and you may try again.

If you forget your password, you will have to contact your System Administrator, who can assign a new password to you. The administrator will NOT be able to tell you your current password, since those are encrypted within the system and are not human-readable.

When you have logged in successfully, the system sets the working directory to your "login home directory," which may be anywhere in your file hierarchy that you please. The login home directory name is first established by the System Administrator for your network when your account is created. You may change it using the -H option (below).

ARGUMENTS

The 'id', 'proj', and 'org' arguments may be separated either by blanks (as shown), or by periods.

id

(required)

Specify the user ID assigned to you by the System Administrator when your access privileges were established.

proj

(optional)

Specify project ID associated with this user id. User IDs may or may not have project IDs, depending on how the access privileges were set up.

org

(optional)

Specify organization ID associated with this user and project ID. Again, this may or may not be necessary for any particular user ID.

OPTIONS

-P

Set new password. After you have successfully logged on with the old password, the system will prompt you to set a new password. The password may be any combination of ASCII characters up to a maximum of 8 characters. (Note: Some network configurations may require the password to be at least 6 characters long for added security. Check with your local System Administrator to see if this requirement has been implemented on your network.)

-H

Set new home directory. After you have logged on successfully, the system will prompt you to establish a new home directory for login. Enter the desired pathname. This is the directory which you enter by default each time you log on.

LO (LOG_OFF)

LO (LOG_OFF) -- Log off a node.

FORMAT

LO [options]

The LO command stops all user processes (except those created by the CPS command and those created by the node startup file 'NODE_DATA/STARTUP), closes all pads and files, and returns the display to the "Please log in:" prompt.

If LO cannot terminate all active processes normally, then the command asks you whether or not you wish to blast the remaining processes (see -F below). Respond either "y" or "n".

You may also disable the ability to log off. See -OFF below.

OPTIONS

-F	Force log off by blasting processes that cannot be stopped normally. If you use this option, be aware that some disk space may be lost. To recover the disk space, use the salvager SALVOL.
-OFF	Disable the ability to log off. When this option is specified, the user who is currently logged in will not be able to log off.
-ON	Enable log off. Use this option to restore the ability for a user to log off.

MSG (MESSAGE) -- Display a message in the DM output window.

FORMAT

MSG 'string'

The MSG command instructs the Display Manager to print a string in the DM output window. The string must be enclosed in single quotes.

ARGUMENTS

string

(required)

Specify the string to be printed in the DM output window.

EXAMPLES

The DM command line:

Command: MSG 'Please select another key'

causes the DM to display the message "Please select another key" in the DM output window.

PB (PAD_BOTTOM)

PB (PAD_BOTTOM) -- Move bottom of pad into window.

FORMAT

PB

The PB command moves the bottom line of the pad to the bottom of the current window. This is a pad movement command, as opposed to TB, which moves the cursor to the last line in the window, regardless of that line's position in the pad.

PB does not require either arguments or options.

PH (PAD_HORIZONTAL) -- Move pad horizontally by characters.

FORMAT

PH [-]n

The PH command moves (scrolls) the pad horizontally under a window in units of characters.

By default, the boxed horizontal arrow keys scroll a pad in 10-character increments.

ARGUMENTS

[-]n (required)

Specify scrolling increment in characters. Positive (unsigned) 'n' scrolls pad left; negative 'n' scrolls pad right.

PN (PAD_NAME)

PN (PAD_NAME) -- Save transcript pad in named file.

FORMAT

PN pathname

The PN command names a transcript pad and makes it permanent. That is, the pad is stored in a file and remains on the system after all windows to it are deleted. The file remains in use and locked, however, until the process is stopped and all windows are closed. If you do not use the PN command, transcript pads are deleted when all windows to them are deleted.

The PN command can also be used to change the name of an edit pad.

ARGUMENTS

pathname (required)

Specify the pathname where the DM saves the pad. The pathname must be cataloged in a directory on your node (i.e., you cannot save a file on your node if the file name is cataloged on some other node).

PP (PAD_PAGE) -- Scroll pad vertically by pages.

FORMAT

PP [-]n

The PP command scrolls the pad vertically under a window in units of pages. By default, the boxed up and down arrow keys invoke this command, scrolling in half-page units.

ARGUMENTS

[-]n (required)

Specify scrolling increment in pages. Positive (unsigned) 'n' scrolls down; negative 'n' scrolls up. Note that 'n' may also be a decimal fraction.

A "page" is defined as the smaller of either of the following values:

- the number of lines that fit in the window
- the number of lines between the bottom of the window and the next form feed or frame

PT (PAD_TOP)

PT (PAD_TOP) -- Move top of pad into window.

FORMAT

PT

The PT command moves the top line of the pad to the top of the current window. This is a pad movement command, as opposed to TT, which moves the cursor to the first line in the window, regardless of that line's position in the pad.

PT does not require either arguments or options.

PV (PAD_VERTICAL) -- Scroll pad vertically by lines.

FORMAT

PV [-]n

The PV command scrolls the pad vertically under a window in units of lines. By default, the shifted up and down arrow keys on the low-profile keyboard and the F2 and F3 function keys on the 880 keyboard invoke this command, scrolling in one line units.

ARGUMENTS

[-]n (required)

Specify scrolling increment in lines. Positive (unsigned) 'n' scrolls down; negative 'n' scrolls up.

PW (PAD_WRITE)

PW (PAD_WRITE) -- Update edit file while maintaining edit pad unchanged.

FORMAT

PW

The PW command updates a file that is being edited. It is valid only for writable edit pads. The first time you issue PW, the DM writes the contents of the edit pad to the file that is being edited, without closing the edit pad. The previous contents of the file are saved in a file with the same name and the added suffix .BAK. Subsequent PW or WC (WINDOW_CLOSE) commands rewrite the new file and leave the .BAK version of the file unchanged.

PW is similar to WC with two exceptions. First, PW leaves the edit pad open. Second, PW writes the new version of the file even if other windows are viewing the edit pad.

PW is useful if, for example, you want to try compiling a program you are editing. If you decide to make additional changes to the program, you can just go back to the edit pad and continue editing, since updates made by PW leave the edit pad open and active.

The <SAVE> key on the low-profile keyboard executes PW;RO to save the pad and put it in read-only mode. There is no predefined key on the 880 keyboard that provides a similar function, although PW is executed along with other DM commands by the default CTRL/Y sequence.

The PW command requires no arguments or options.

RM (REPLACE_MARK) -- Replace a mark on the mark stack.

FORMAT

RM

The RM command places the last issued mark (DR) back on the mark stack, allowing you to use the mark again.

RM requires no arguments or options.

RO (READ_ONLY)

RO (READ_ONLY) -- Set read/write mode.

FORMAT

RO [-ON | -OFF]

The RO command puts a pad into (-ON) and out of (-OFF) read-only mode. If no option is supplied, the current mode is toggled. The pad must be in write mode (-OFF) in order for you to insert or delete anything.

By default, CTRL/M invokes the RO command without options to toggle the current mode.

An "R" appears in the window legend of a pad in read-only mode. The "R" disappears in write mode.

An edit pad which has been modified cannot be made read-only without first writing it out with the PW command. See the PW command description for details.

RS (REFRESH_SCREEN) -- Refresh screen.

FORMAT

RS

The RS command refreshes the entire screen, updating all windows with any pending changes. By default, the CTRL/F sequence invokes this command.

RW (REFRESH_WINDOW)

RW (REFRESH_WINDOW) -- Refresh a window.

FORMAT

RW [-R]

The RW command causes the DM to refresh the contents of the current window immediately, updating it with any pending changes.

When an unexpected system fault, such as a network failure, occurs, pads may be marked undisplayable in order to avoid further faults. When this happens, the DM displays an error message instead of the window's normal contents. When the problem has been resolved, use the -R option (reset) to redisplay the window's normal contents.

S (SUBSTITUTE) -- Substitute all occurrences of matched string in defined range.

FORMAT

[range]S[[/[string1]]/string2/]

The S command substitutes one literal string for a string described by a regular expression over a defined text range. The command does not move the cursor or the window, but does update the window when the substitution is completed. Strings used with this command are also saved for later use (see below).

All substitutions are case sensitive, unlike searches, which ignore case unless told otherwise. Substitution case sensitivity cannot be disabled.

If the Display Manager scans more than 100 lines while processing a substitute command, it displays a "Substitute in progress..." message in its message window. Then it polls for keystrokes for every 10 lines it processes. At this point you may:

- 1. Wait for DM to complete the operation.
- 2. Use the keyboard. It works as it does normally. You can type into any pad except the one being searched. You can issue any Display Manager command except another search or substitute command. The Display Manager executes these commands when it completes the substitution. You can type input to another Shell or program (if it was previously waiting for input). Subsequent user process input and output requests will be processed when the Display Manager finishes the substitution.

ARGUMENTS

If no arguments are specified, the previous substitution will be repeated from the current cursor position to the end of the line.

range

(optional)

Specify range of text in which substitution is to be made. See the section on defining text ranges in the previous chapter for details.

Default if omitted: use current cursor position to end of line

string1

(optional)

Specify string to be replaced in the form of a regular expression. If this argument is omitted but the opening delimiter (/) is used (i.e., "S//string2/"), then string1 defaults to the string1 used in the last search operation. If the delimiter is also omitted (i.e., "S/string2/"), then string1 defaults to the string used in the last substitution operation.

Default if omitted: see above

S (SUBSTITUTE)

string2

(optional)

Specify literal replacement string. This is not a regular expression). An "&" can be used to denote string1. If 'string1' is present, then 'string2' is required.

Default if omitted: repeat last substitution command

EXAMPLES

Move to first character in the pad.
Place a mark.
Move to last character in the pad.
Replace the string "Fielding" with
"Tom Jones" throughout the marked
range (in this case, the entire pad).

2. <CMD> s/Tom/& Jones/

Replace "Tom" with "Tom Jones".

Since no range was marked or specified, the replacement takes effect from the current cursor position to the end of the line.

See the section on "Using Regular Expressions" in the previous chapter for more examples.

SC (SET CASE) -- Set search case sensitivity.

FORMAT

SC [-ON | -OFF]

A search can be either case-sensitive or case-insensitive. In case-sensitive searches, the characters must match in case (i.e., /mary/ would NOT locate the string "MARY"). In case-insensitive searches, uppercase and lowercase letters are considered equivalent. By default, searches are case-insensitive.

The -ON option explicitly specifies a case-sensitive search; the -OFF switch explicitly specifies a case-insensitive search. Typing the SC command without options toggles the current case comparison setting.

NOTE: The SC command has no effect on *substitution* operations, only *search* operations. Substitutions are always sensitive to the case of the strings involved.

SHUT -- Shut down system.

FORMAT

SHUT [-F]

The SHUT command exits from the Display Manager and shuts down the system. The Display Manager first closes all windows and pads, then unloads the operating system (AEGIS) and enters the Mnemonic Debugger that resides in the node's boot PROM. If user processes are still active, the SHUT command attempts to stop them. If they stop normally, the shutdown proceeds. If the Display Manager cannot stop them normally, the SHUT command aborts.

The SHUT command has the same effect on system software as turning the node's power off.

To restart the system, type EX AEGIS in the Mnemonic Debugger.

To force either log off or shut down, specify the -F option. (The same effect can be achieved by reponding "y" to a request to blast processes that cannot be closed normally.) If you use this, however, be aware that some disk space may be lost if processes cannot be terminated normally. To recover the disk space, use the salvager SALVOL. See the SALVOL (SALVAGE_VOLUME) command.

SO (SUBSTITUTE_ONCE) -- Substitute first occurrence of matched string.

FORMAT

[range]SO[[/[string1]]/string2]

The SO command is identical to the S (SUBSTITUTE) command except that 'string2' replaces only the first occurrence of 'string1' in each line of the defined range of text.

SQ (SEARCH_QUIT)

SQ (SEARCH_QUIT) -- Abort a search operation.

FORMAT

SQ

The SQ command aborts a text search, and cancels any action involving the ECHO command. This command is equivalent to ABRT.

The SQ command aborts the current search. The DM returns the message "Search aborted." It does not move the window. Note that you cannot type this command during a search. You must invoke it with a defined key.

When you use SQ to abort the ECHO command, SQ cancels a 'move window with rubberbanding' or 'grow window with rubberbanding' operation; or, it cancels highlighting for a defined range of text, depending on how ECHO was used.

TB (TO_BOTTOM) -- Move cursor to bottom line in window.

FORMAT

TB

The TB command moves the cursor to the bottom line in the window. This is in contrast to the PB command, which moves the bottom line of the pad into the window.

TB requires no arguments or options.

NOTE:

There is a homonymous Shell command: TB (TRACEBACK) -- Print traceback after a fault. See the TB command description in the Shell chapter for details.

TDM (TO_DM_WINDOW)

TDM (TO_DM_WINDOW) -- Move cursor to DM input window.

FORMAT

TDM

TDM moves the cursor to the Display Manager input window (labeled "Command: " at the bottom of the screen) so that you can enter DM commands.

By default, the <CMD> key (L5) invokes the TDM command.

TDM requires no arguments or options.

TH (TAB_HORIZONTAL) -- Move cursor right to next tab stop.

FORMAT

TH

The TH command moves the cursor right to the next horizontal tab stop. Tabs are global (i.e., they apply to all windows), and may be set with the DM command TS. Initially, tabs are set every 5 spaces.

By default, the <TAB> key invokes the TH command. Note that this does NOT insert an ASCII tab character into the file; it simply positions the cursor to the next tab stop.

TH requires no arguments or options.

THL (TAB_HORIZONTAL_LEFT)

THL (TAB_HORIZONTAL_LEFT) -- Move cursor left to previous tab stop.

FORMAT

THL

The THL command moves the cursor left to the next horizontal tab stop. Tabs are global (i.e., they apply to all windows), and may be set with the DM command TS. Initially, tabs are set every 5 spaces.

By default, the CTRL/<TAB> sequence invokes the THL command. Note that this does NOT insert an ASCII tab character into the file; it simply positions the cursor to the previous tab stop.

THL requires no arguments or options.

TI (TO_INPUT_WINDOW) -- Move cursor to next input window.

FORMAT

TI

TI moves the cursor to the next fully unobscured window in which input is accepted (i.e., the next window that opens into neither a transcript nor a read-only edit pad). The cursor is placed at its last previous position in the window.

The Display Manager scans across the screen from left to right and top to bottom to find the next window.

TL (TO_LEFT)

TL (TO_LEFT) -- Move cursor to the beginning of the current line.

FORMAT

TL

The TL command moves the cursor left to the beginning of the current line. By default, the bar-left arrow key invokes the TL command.

TL requires no arguments or options.

TLW (TO_LAST_WINDOW) -- Move cursor to last (previous) window.

FORMAT

TLW

TLW moves the cursor back to the window it was in before it moved to the current window. The cursor is placed at its last previous position in the window.

By default, the CTRL/L sequence invokes the TLW command.

TN (TO_NEXT_WINDOW)

TN (TO_NEXT_WINDOW) -- Move cursor to next window.

FORMAT

TN

TN moves the cursor to the next fully unobscured window on the screen. Any window that is partially covered by another is not considered in the search. The DM scans the screen from top to bottom to find the next window, selecting the one whose upper-left corner is the "highest" (i.e., has the lowest Y coordinate value), then proceeding downward across the screen. If there are panes within a window, the cursor is positioned in the next lower pane until the pane choices have been exhausted, before moving to the next "lower" window. Once the next window is located, the DM places the cursor at its last previous position within that window.

By default, the <NEXT WNDW> key (LB) invokes this command.

TNI (TO_NEXT_ICON) -- Move cursor to next icon.

FORMAT

TNI

TNI moves the cursor to the next fully unobscured icon on the screen. Any icon that is partially covered by another is not considered in the search. The DM scans the screen from top to bottom to find the next icon, selecting the one whose upper-left corner is the "highest" (i.e., has the lowest Y coordinate value), then proceeding downward across the screen.

This command is similar to the TN command, which positions the cursor to the next fully unobscured window on the screen.

TR (TO_RIGHT)

TR (TO_RIGHT) -- Move cursor to the end of the current line.

FORMAT

TR

The TR command moves the cursor right to the end of the current line. By default, the bar-right arrow key invokes the TR command.

TR requires no arguments or options.

TS (TAB_SET) -- Set tab stops for all windows.

FORMAT

The TS command sets the default tab stops for all windows. Tab stops may also be set from within a program using a call to the system routine PAD_\$SET_TABS; those set under program control override the tab stops set by TS within windows belonging to the program.

By default, tabs are initially set every 5 spaces.

NOTE: The DM command "=" displays the line and column numbers of the current cursor position. This can be helpful when trying to set tab stops visually.

ARGUMENTS

If no arguments are specified, a stop is set at every character on the line.

n1 n2 ...

(optional)

Specify tab stops. The 'n' values are integers representing absolute character positions. They must appear in increasing

order. Columns are numbered starting with one.

Default if omitted: see above

OPTIONS

-R

Repeat the last interval.

EXAMPLES

Command: TS 7 12 -r

(Set tabs at columns 7 and 12, and every 5 spaces thereafter.)

TT (TO_TOP)

TT (TO_TOP) -- Move cursor to top line in window.

FORMAT

TT

The TT command moves the cursor to the top line in the window. This is in contrast to the PT command, which moves the top line of the pad to the top of the window.

TT requires no arguments or options.

TWB (TO_WINDOW_BORDER) -- Move cursor to a specified window border.

FORMAT

The TWB command moves the cursor to a border of the current window, as specified by the command options. You must specify an option with TWB.

OPTIONS

One of the following options is required.

-L	Move the cursor to the left window border parallel to the previous cursor position.
-R	Move the cursor to the right window border parallel to the previous cursor position.
- T	Move the cursor to the top window border directly above the previous cursor position.
- B	Move the cursor to the bottom window border directly below the previous cursor position.

UNDO -- Undo previous DM command(s).

FORMAT

UNDO

UNDO works by compiling a history of DM activities in input and edit pads in reverse chronological order. Invoking UNDO reverses the effect of the most recent DM command. Successive UNDOs will undo further back in history. Note that this only applies to DM operations; Shell operations (such as compiling a program) cannot be undone.

The UNDO buffers (one per edit pad and one per input pad) are circular lists that, when full, eliminate the oldest entries to make room for new ones. Entries are grouped together in sets. For example, a S (SUBSTITUTE) command may change five lines. While UNDO considers this to be five entries, the five entries are grouped into a single set so that one UNDO will change all five lines back to their original state. When a buffer becomes full, the oldest set of entries is erased. This means that UNDO will never partially undo an operation: it will either completely undo it or do nothing.

An edit undo buffer can hold up to 1024 entries. An input undo buffer can hold up to 128 entries.

By default, the <UNDO> key on low-profile keyboards invokes the UNDO command. There is no predefined key for this function on 880 keyboards.

UNDO requires no arguments or options.

WA (WINDOW_AUTOHOLD) -- Set window autohold mode.

FORMAT

WA [-ON | -OFF]

NOTE: Autohold mode applies only to windows open into transcript pads.

The WA command switches a window into (-ON) and out of (-OFF) autohold mode. WA without options toggles the current setting. In autohold mode, the window automatically enters hold mode (in which the contents of a window are temporarily frozen) if either of the following conditions is true:

- A full window of output is available and none of it has been displayed.
- A form feed or create frame operation is output to the pad. In this case, the window displays the output preceding the form feed. When the window exits from hold mode, the output following the form feed or create frame operation starts at the top of the window.

Initially, windows are not in autohold mode. The window legend contains an "A" when the window is in autohold mode.

WC (WINDOW_CLOSE) -- Close window and associated functions.

FORMAT

The WC command closes (deletes) a window. It may also close the pad into which the window looks, depending on the following conditions.

If other windows into the pad besides the one being closed exist, the DM naturally leaves the pad open. If there are no other windows into the pad, however, the DM closes it. The closed pad is then either deleted (if it was temporary) or saved under its pathname (if it was named and permanent, i.e., a permanent disk file).

If the pad is a writable edit pad, and is being viewed only through the current window, the DM renames the old file by appending .BAK to its name, and writes the edited version to the original file name. If multiple windows are viewing the edit pad, WC simply closes the window -- it does not write the file or rename the old file. To force the DM to write the file and create the .BAK version, use the DM command PW (PAD_WRITE) or the <EXIT> or CTRL/Y keys (see below).

A transcript (output) window normally cannot be closed if it is the last window into an active process (see -F below).

Note that the DM cannot delete a permanent pad (file); you must use the Shell command DLF (DELETE_FILE) for this purpose.

Two keys (or control/key sequences) have been predefined to perform related functions:

		KBD 2	KBD 1	
PW;WC -Q	(or)	<exit> (R5)</exit>	CTRL/Y	Close window, pad; update file
WC -Q	(or)	<abort> (R5S)</abort>	CTRL/N	Close window, pad; ignore changes

OPTIONS

If no options are specified, WC closes the window and pad, then deletes the pad (if temporary) or rewrites it (if permanent) as described above. Only one of the following options may be specified at a time. Auto-close is disabled by default.

-Q	Quit without updating pad (file). Any changes made while
	window was open are ignored. The system prompts you with
	"File modified. OK to quit?" if you have made changes to verify
	that you really wish to discard them.

-F Force window closure, even if this window was the last one open into a process. Note that the process will become inaccessible, however, if no windows are left.

-A

Enable auto-close for current window. When auto-close is enabled, the current window will close when the pad into which it looks is closed.

-S

Disable auto-close for current window. If auto-close is disabled (the default condition), then the current window persists after the pad into which it looks is closed.

WDF (WINDOW_DEFAULT) -- Define DM default window positions.

FORMAT

[region]WDF [n]

The WDF command lets you define any of the DM's five default window positions. To define a default window postion, mark (with the DM command "DR") the region that will display the window, and issue the WDF command.

ARGUMENTS

region

(optional)

Specify the area of the screen where the new window will be displayed. For details on window regions, see "Defining Points and Regions" elsewhere in this chapter.

Default if omitted: use marked region

n

(optional)

Specify the ID number (1-5) of the DM default window that is being defined. If you omit n, the WDF command discards any saved window parameters, so the next window created uses the stock default window boundaries.

Default if omitted: see above

WG (WINDOW_GROW) -- Grow or shrink a window.

FORMAT

[region]WG

NOTE: There is a companion grow command WGE that provides visible feedback during a grow operation. See the WGE command description elsewhere in this chapter for information on that command.

The WG command changes the size of a window by moving one edge or corner across the screen while leaving the other edges and/or corners where they are. To grow or shrink a window, first mark the edge or corner you want to move by positioning the cursor at that edge or corner and issuing the DR command string or pressing <MARK>. Then move the cursor to the edge or corner's new location and issue the WG command. (By default, CTRL/G invokes the WG command on low-profile keyboards. This function is not available by default on 880 keyboards.) The marked edge or corner moves to the new cursor position, and the window shrinks or grows accordingly. If you want to move only an edge, move the cursor only in the direction perpendicular to that edge. Moving the cursor in two dimensions causes a corner to move.

ARGUMENTS

region (optional)

Specify old and new locations of edge or corner. May be specified in a variety of formats: see "Defining Points and Regions" in the previous chapter. This argument is REQUIRED if you do not use the cursor placement and <MARK> operation described above.

Default if omitted: use marked region

WGE (WINDOW_GROW_ECHO)

WGE (WINDOW_GROW_ECHO) -- Grow/shrink a window with rubberbanding.

FORMAT

WGE

The WGE command changes the size of a window. To enlarge or shrink a window with WGE, position the cursor in the window and issue the WGE command. (By default, the <GROW> key invokes the WGE command on the low-profile keyboard, while CTRL/G provides the same function on the 880 keyboard.) After you enter the WGE command, an outline, or "rubberband" will appear to show you the size and shape that the window will take when you complete the grow operation. Move the cursor until the rubberband matches the new size you want for the window. Then issue the DR; ECHO command sequence or press <MARK> to complete the grow operation. You can use the SQ command (CTRL/X) to abort a grow operation using rubberbanding.

WGE requires no arguments or options.

WGRA (WINDOW_GROUP_ADD) -- Create or add to a window group.

FORMAT

WGRA group _ name [entry _ name]

The WGRA command creates a new window group with the specified group_name, or adds a window or group to an existing group. By default, if you do not specify an entry_name (the name of a window or group) WGRA uses the pathname of the window where the cursor was last positioned.

ARGUMENTS

group_name

(required)

Specify the name of the group to be created or enlarged.

entry_name

(optional)

Specify the name of the window or group to be added to

'group_name'.

Default if omitted: Use the pathname of the window where the

cursor was last positioned.

EXAMPLES

Command: WGRA Shell Windows Process_1

This command adds a window called "Process_1" to a group of windows called "Shell Windows".

WGRR (WINDOW_GROUP_REMOVE)

WGRR (WINDOW_GROUP_REMOVE) -- Remove window/group from group.

FORMAT

WGRR group_name [entry_name]

The WGRR command removes a window or group from a window group. By default, if you do not specify an entry_name (the name of a window or group) WGRR uses the pathname of the window where the cursor was last positioned.

ARGUMENTS

group_name

(required)

Specify the name of the window group that contains the window

or group you want to remove.

entry_name

(optional)

Specify the name of the window or group to be removed.

Default if omitted: use the pathname of the window where the

cursor was last positioned

EXAMPLES

Command: WGRR Shell_Windows Process_2

This command removes a window called "Process_2" from a group of windows called "Shell_Windows".

WH (WINDOW_HOLD) -- Set window hold mode.

FORMAT

WH [-ON | -OFF]

NOTE: Hold mode applies only to windows open into transcript pads.

The WH command switches a window into (-ON) or out of (-OFF) hold mode. WH without options toggles the current setting. In hold mode, the contents of the window are "frozen" and do not change when a program sends more output to the pad. When a window is not in hold mode, the window automatically moves to the end of the pad as new output appears.

By default, the <HOLD> key on low-profile keyboards and the <HOLD/GO> key on 880 keyboards invoke the WH command.

Initially, windows are not in hold mode. The window legend contains an "H" when the window is in hold mode.

WI (WINDOW_INVISIBLE)

WI (WINDOW_INVISIBLE) -- Make a window or group visible or invisible.

FORMAT

WI [entry_name] [-W | -I]

The WI command controls the visibility or invisibility of the specified window or group. WI without options toggles the current mode. By default, if you do not specify an entry_name (the name of a window or group) WI uses the pathname of the window where the cursor was last positioned.

ARGUMENTS

entry_name

(optional)

Specify the name of the window or group you want to make

visible or invisible.

Default if omitted: use the pathname of the window where the

cursor was last positioned

OPTIONS

If no option is specified, WI toggles the current visiblity setting.

-I

Force the window or group to be invisible.

-W

Force the window or group to appear as a window.

WM (WINDOW_MOVE) -- Move a window across the screen.

FORMAT

[region]WM

NOTE: There is a companion move command WME that provides visible feedback during a move operation. See the WME command description elsewhere in this chapter for information on that command.

The WM command moves a window across the screen. The DM moves the window whose nearest unobscured edge or corner is the first point of the region. The new location of this edge or corner is the second point of the region. So, to move a window, place the cursor at one corner of the window you want to move and issue the DR command (or press <MARK>). Next, place the cursor at the desired new position of the corner. Finally, issue the WM command.

If you do not define a region, the WM command causes the nearest window corner to be moved to the current cursor position. This can cause unexpected results if there are multiple windows on the screen, however, since your idea of the "nearest" window may not be the same as the DM's. In that case, it is safer to <MARK> the window you want to move.

ARGUMENTS

region

(optional)

Specify old and new locations of edge or corner. This may be specified in a variety of formats: see "Defining Points and Regions" in the previous chapter.

Default if omitted: use current cursor position

WME (WINDOW_MOVE_ECHO)

WME (WINDOW_MOVE_ECHO) -- Move a window using rubberbanding.

FORMAT

WME

The WME command moves a window across the screen using the rubberbanding feature. To move a window, place the cursor in the window you want to move and issue the WME command. (By default, the <MOVE> key invokes the WM command on the low-profile keyboard while CTRL/W provides the same function on the 880 keyboard.) After you issue the WME command an outline or "rubberband" will appear to show you where the window will be when you complete the move operation. Now move the cursor until the rubberband is at the desired window position. Finally, issue the DR; ECHO command or press <MARK> to complete the grow operation. You can use the SQ command (CTRL/X) to abort a move operation using rubberbanding.

The WME command requires no arguments or options.

WP (WINDOW_POP) -- Push or pop a window on the stack.

FORMAT

WP [window_name|group_name] [-T|-B]

The WP command pops a window to the top of the stack or pushes a window to the bottom of the stack. If the cursor rests in a partially obscured window, the WP command pops the window to the top of the pile. If the cursor rests in a completely visible window, WP pushes the window to the bottom of the pile. WP can also manipulate specific named windows or window groups. See the ARGUMENTS section below.

By default, the WP command is invoked by the <POP> key on low-profile keyboards, while the same function is provided by CTRL/P on 880 keyboards.

ARGUMENTS

window_name

(optional) Push or pop the window with the specified name. Not valid if

'group _ name' is specified.

Default if omitted: Push or pop the window containing the

cursor.

group_name

(optional) Push or pop all windows with the specified group. Not valid if

'window_name' is specified.

Default if omitted: Push or pop the window containing the

cursor.

OPTIONS

The following options are intended primarily for use in DM scripts, where you may not be able to predict the presence of other windows on the screen.

T Force a window to the top of the window stack.

-B Force a window to the bottom of the window stack.

EXAMPLES

Command: wp -t Pop the window containing the cursor

to the top of the window stack.

Command: wp slide -b Push the window named 'slide' to the

bottom of the stack.

WS (WINDOW_SCROLL)

WS (WINDOW_SCROLL) -- Set window scroll mode.

FORMAT

WS [-ON | -OFF]

NOTE: Scroll mode applies only to windows open into transcript pads.

The WS command switches a window into (-ON) and out of (-OFF) "line-at-a-time" scrolling. WS without options toggles the current setting. When "line-at-a-time" scrolling is in effect, output appears in the window one line at a time, scrolling past. When "line-at-a-time" scrolling is not in effect, output appears a window at a time.

By default, CTRL/S invokes the WS command.

Initially, all windows (except edit windows) have "line-at-a-time" scrolling. The window legend contains an "S" when the window is in scroll mode.

XC (COPY) -- Copy text to paste buffer.

FORMAT

[range]XC [-R] [-F pathname | name]

The XC command copies a range of text from any pad into a paste buffer or system file. The copied text remains undisturbed.

By default, the <COPY> key on low-profile keyboards and CTRL/C on 880 keyboards invoke the XC command using the default (unnamed) paste buffer.

ARGUMENTS

range

(required) Specify range of text to be copied. Define the range to be

copied as described in "Defining a Range of Text" in the

previous chapter.

Default if omitted: copy from cursor to end of line

name

(optional) Specify paste buffer name. Text is written to the named buffer.

If text is copied to a buffer that has previously been used, the new text overwrites the old. You may have up to 100 buffers

open per log in session.

Default if omitted: use unnamed buffer

OPTIONS

-F pathname Specify system file to receive copied text. If the file already

exists, the copied text overwrites the current file contents. Not

valid if 'name' argument is present.

-R Specify copy for a rectangular portion of text.

XD (CUT) -- Cut (delete) text and write it to paste buffer.

FORMAT

[range]XD [-R] [-F pathname | name]

The XD command copies a range of text into a paste buffer or system file, then deletes the text from the pad. This command can be used only in a writable pad.

By default, the <CUT> key on low-profile keyboards and CTRL/E on 880 keyboards invoke the XD command using the default (unnamed) paste buffer.

ARGUMENTS

range

(required)

Specify range of text to be cut. Define the range to be cut as described in "Defining a Range of Text" in the previous

chapter.

Default if omitted: cut from cursor to end of line

name

(optional)

Specify paste buffer name. Text is written to the named buffer. If text is copied to a buffer that has previously been used, the new text overwrites the old. You may have up to 100 buffers open per log in session.

Default if omitted: use unnamed buffer

OPTIONS

-F pathname

Specify system file to receive cut text. If the file already exists, the cut text overwrites the current file contents. Not valid if

'name' argument is present.

-R

Specify cut for a rectangular portion of text.

XI (COPY_IMAGE) -- Copy a display image into a graphics map file.

FORMAT

[range]XI [-F pathname]

The XI command copies a display image into a graphics map file (GMF). If you do not mark the portion of the display window you want to copy, XI copies the entire window where the cursor is positioned. You can print the GMF using the PRF Shell command with the -PLOT option.

ARGUMENTS

range

(optional)

Specify range of image to be copied. Define the range to be copied as described in "Defining a Range of Text" in the

previous chapter.

Default if omitted: copy current window

OPTIONS

-F pathname

Specify GMF output file. If this option is omitted, the image is written to 'NODE_DATA/PASTE_BUFFERS/DEFAULT.GMF. You can print the GMF using the PRF Shell command with the -PLOT option.

XP (PASTE) -- Paste (write) buffered text into pad.

FORMAT

XP [-R] [-F pathname | name]

The XP command inserts the contents of a paste buffer or system file into a pad at the current cursor position. The contents of the paste buffer or file are unchanged by this command, making multiple insertions possible. This command can be used only in a writable pad.

By default, the <PASTE> key on low-profile keyboards and CTRL/O on 880 keyboards invoke the XP command using the default (unnamed) paste buffer.

ARGUMENTS

name

(optional)

Specify paste buffer name. Text is copied from the named buffer. You may have up to 100 buffers open per log in session.

Default if omitted: use unnamed buffer

OPTIONS

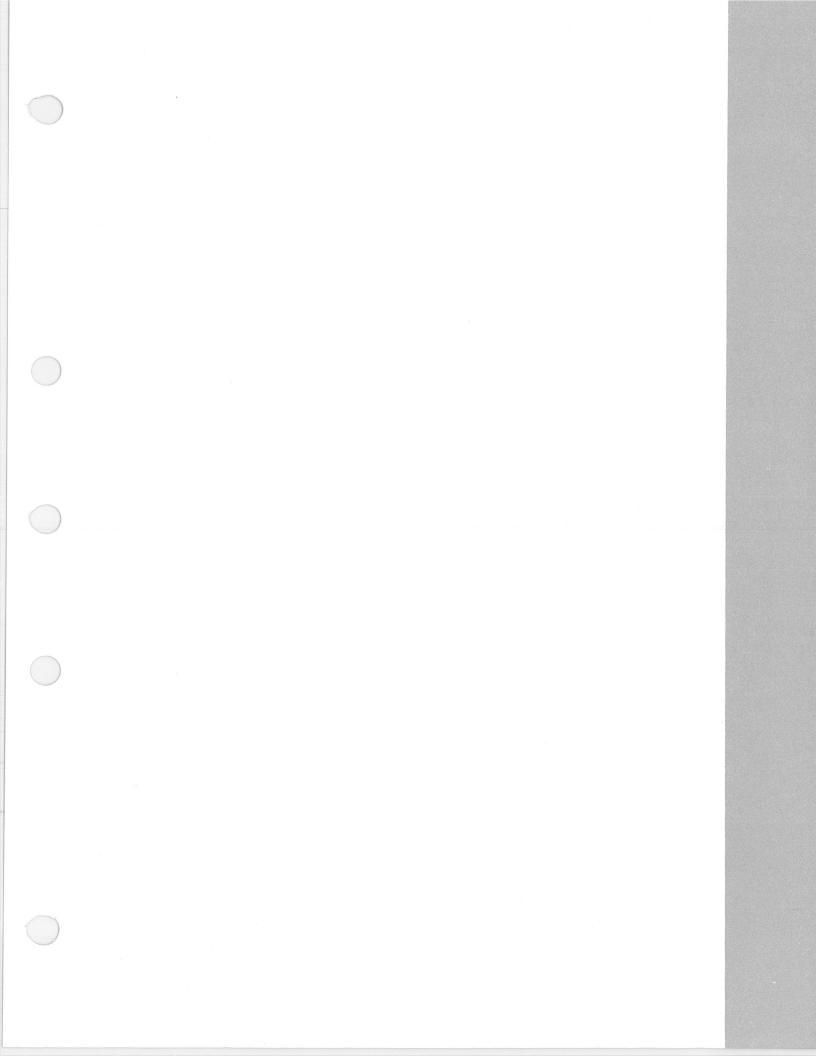
-F pathname

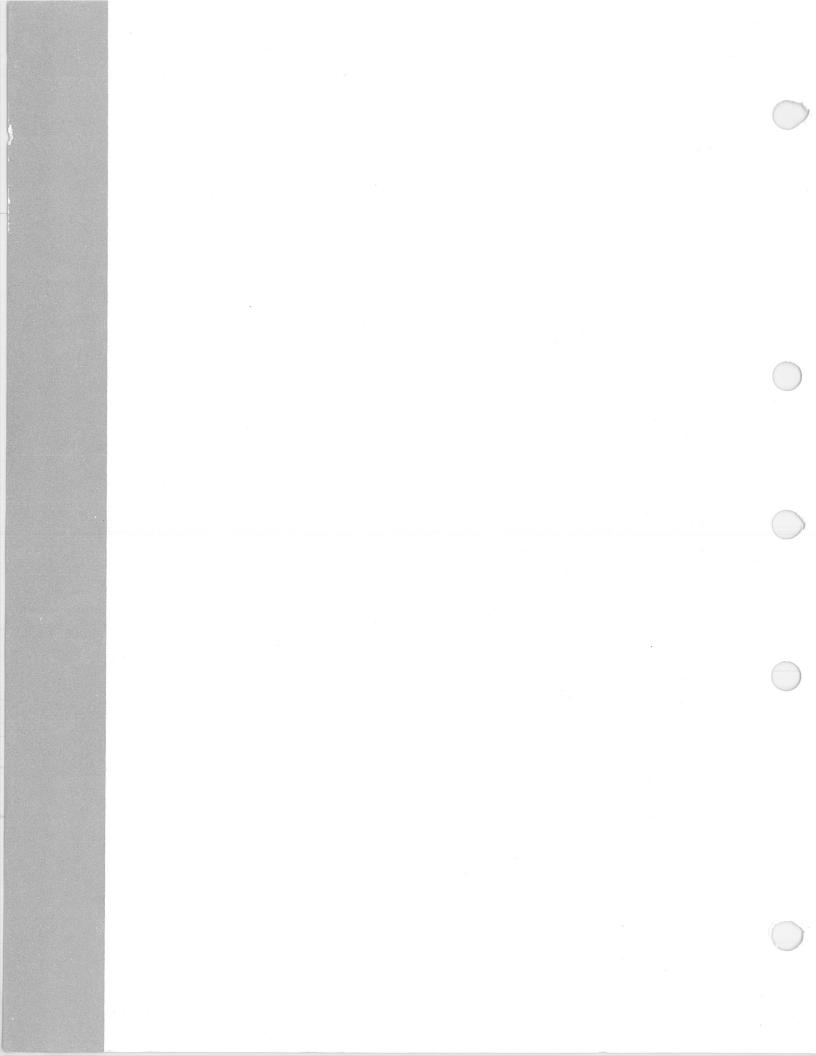
Specify system file to provide paste text. Not valid if 'name'

argument is present.

-R

Specify paste of a rectangular portion of text.





Chapter 3 Shell Basics

This chapter summarizes the basic concepts that apply to the Shell commands described individually in the following chapter. For a more indepth discussion of these concepts, please refer to the DOMAIN System User's Guide.

3.1. Command Format

In the most general sense, the operating system has no commands. There are simply files that the Shell looks for and executes. When you type "date" in the Shell input window, the Shell looks for a file called DATE (following its command search rules) and executes it. This means that any files that you create can be given to the Shell for execution. (Of course, if you tell the Shell to execute a file containing non-binary data — like the text of a memo — you will receive an error message.) The point is that any file, no matter where it comes from, may be given to the Shell for interpretation and execution.

The simplest command line consists of a command name followed by arguments to the command, separated by spaces:

```
$ command arg1 arg2 ... argn <RETURN>
```

Normally, you enter one command per line. You may continue a command over several lines by typing @<RETURN> at the end of each line to be continued. The Shell then prompts with "\$ " to indicate that the current line continues the previous one.

3.1.1. Arguments

The command Shell, which we supply, handles commands that accept multiple arguments (Figure 3-1). Usually, those arguments come in two forms: a pathname designating a file on which to operate or some other sort of literal string for manipulation, and instructions for special command action. Those arguments that specify special action are almost always optional, and are immediately preceded by a hyphen (-). (The hyphen is necessary because these arguments often require secondary arguments of their own. The commands use the hyphen to interpret correctly where all the different arguments apply.) These special arguments are labeled "Options" in the command descriptions in Chapter 4.

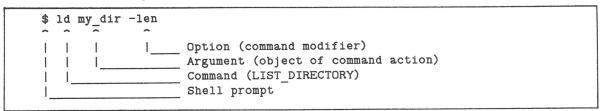


Figure 3-1. Typical Shell Command Format

3.1.2. Separators

Normally, Shell commands are separated from each other by carriage returns (NEWLINE characters). You may place multiple commands on the same line by separating the commands with semicolons, up to a total of 256 characters per line. For example,

\$ wd //mydir/sub1;ld

This command line sets your working directory to the directory "//mydir/sub1" and then lists the contents of that directory.

Multiple commands may also appear on a single line when you are using pipes and filters.

3.2. Using Special Characters

The Shell recognizes a variety of special characters that allow you to change the action of commands. The characters in Table 3-1 have special meanings when they appear on a command line. Note that, while some of these characters have already been discussed as having special meanings in Display Manager commands, regular expressions, and so forth, those meanings do not necessarily carry over to the command Shell environment. Please be careful to keep the different meanings distinct: regular expressions appearing in Shell commands, for instance, should be enclosed in quotation marks to avoid confusion.

The at sign (@) is the Shell's **escape character**. You can place an "@" anywhere on the command line to suppress the special meaning of the next character (including the "@" character itself).

For a full discussion of the usage of Shell special characters, please refer to the DOMAIN System User's Guide.

Table 3-1. Command Shell Special Characters

Character	Usage
?	Match any single character except NEWLINE
%	Match zero or more characters up to but not including period
*	Match zero or more occurrences of preceding character
[string]	Match any single character in character class "string"
[~string]	Match any character except those in "string"
	Match zero or more subordinate directories
= '' '' '' ''	Copy (derive) leafname from previous argument
(names)	Group pathnames for use in later derived names
{expr}	Tag expression for later use

3.3. The Command Line Parser

Many Shell commands that we supply share a standard command line parsing procedure. It determines how each command processes command line information. Chapter 4 of this manual, and the on-line HELP files, identify commands that use the command line parser. These commands support the following features:

- 1. You may use wildcards to specify existing pathnames.
- 2. You may use derived names to specify logically-related pathnames, and parentheses to create several derived names with one command line. (See Table 3-1.)
- 3. When pertinent, you may include multiple pathnames as command line arguments. For example, "PRF file1 file2 file3".
- 4. You may use the asterisk character (*) to cause commands to read pathnames from standard input or from another file. For example,

\$ prf */fred/names_file

prints the files listed in /FRED/NAMES_FILE. Also,

Table 3-1. Command Shell Special Characters (cont.)

Input/Output Control

Character	Usage	Notes
<	Redirect standard input	(3)
</td <td>Redirect error input</td> <td>(3)</td>	Redirect error input	(3)
< </td <td>Read in-line data from standard input</td> <td>(3)</td>	Read in-line data from standard input	(3)
< /</td <td>Read in-line data from error input</td> <td>(3)</td>	Read in-line data from error input	(3)
>	Redirect standard output	(3)
>?	Redirect error output	(3)
>>	Append standard output	(3)
>>?	Append error output	(3)
1	Pipe standard output	(1)
()	Group commands for I/O redirection	(1)

Parsing Operators

Character	Usage	Notes
#	Comment line in a command file	(4)
æ	Run a program or command in the background	(1)
^	Insert parameter	(3)
!	Insert parameter and rescan	(3)
~"cmd"	Insert output of "cmd", with expansion	(3)
~,cmd,	Insert output of "cmd", no expansion	(3)
;	Separate commands on a line	(1)
H	Quoted string, with expansion	(4)
,	Quoted string, no expansion	(4)
0	Escape character	(5)
	Space	(2)

Notes:

- (1) Special anywhere; causes a new command to start
- (2) Special anywhere; causes a new argument to start
- (3) Special anywhere; does not start a new argument
- (4) Special only at the beginning of an argument
- (5) Special only when immediately preceding a character that would otherwise be special

```
$ PRF *
  file1
  file2
  file3
  ***EOF***
```

reads the names "file1", "file2" and "file3" from standard input, and prints each file. When using the keyboard for standard input, a NEWLINE and an end-of-file character must follow the last name. By default, CTRL/Z generates an end-of-file character.

If you include more than one name on an input line, in standard input or in a names file, the command interprets all names except the first one on each line as derived names. For example

\$ CHN *	is equivalent to	\$ CHN * = .old
a a.old		a
b b.old		Ъ
c c.old		С
EOF		**E0F***
\$		\$

Do not confuse the action of the "*" character with that of the input redirection symbol "<". The "*" character causes a Shell command to read pathnames from standard input or from another file. The "<" symbol causes a Shell command to read data from a file.

3.3.1. Standard Command Options

All Shell commands that we supply support the following standard options:

-HELP Display detailed usage information.

-USAGE Display brief usage summary.

-VERSION Display software version number.

NOTE: Using any of these three standard options precludes using any other options within the same command.

3.3.2. Command Line Parser Options

Commands that use the command line parser also support the following options (a "(D)" indicates a default option):

-AE Abort if a name in pathname cannot be found. If omitted, processing continues to next name.

-NQ (D) Do not issue query to verify wildcard names.

-QW Issue query to verify wildcard names.

-QA Issue query to verify all names.

- (hyphen alone) Read further data from standard input. End input with CTRL/Z.

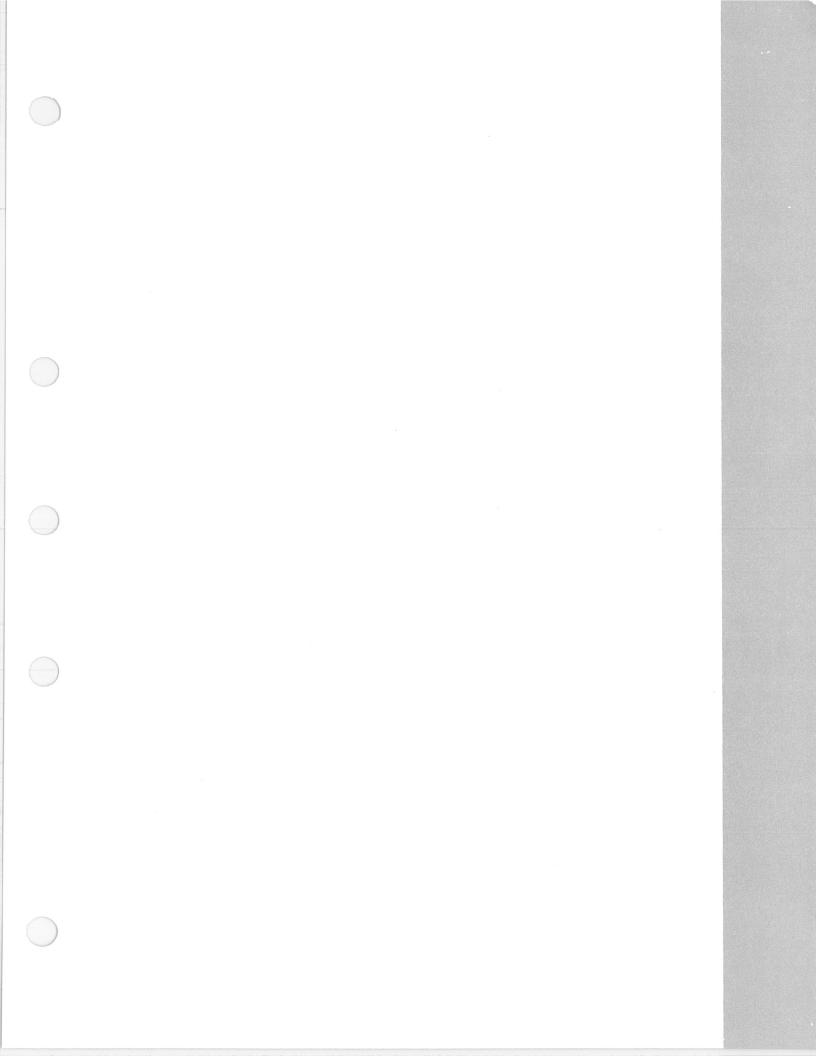
*[pathname] Read file specified for further pathname arguments. If pathname is omitted, read standard input for further pathname arguments.

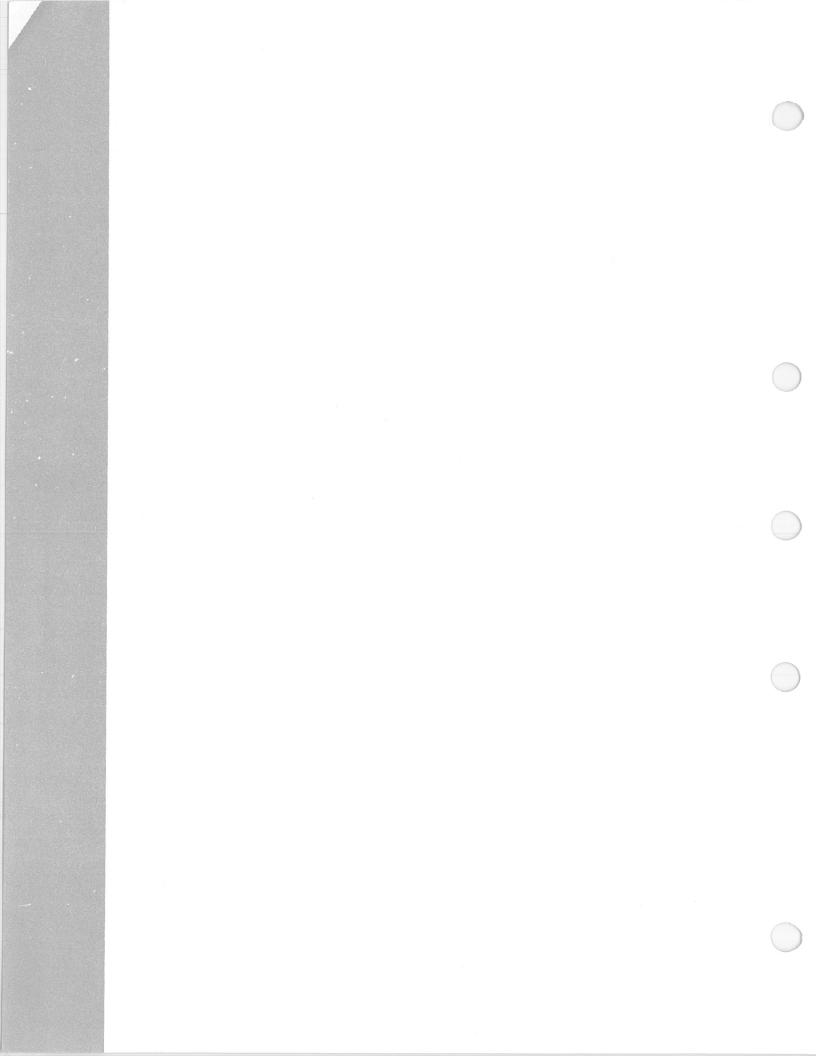
Commands that delete or modify objects automatically verify names specified with wildcards. You can suppress this query using -NQ, or extend it to all names using -QA.

When you select a query option, the command writes the selected names to the error output stream with a? to prompt you for a response. Then it reads your response from the error input stream (normally the keyboard).

If you respond:	The command:
h	displays help information
y	operates on the name
n	ignores the name
q	quits immediately
g	operates on the name and suppresses further name queries
d new_default	resets the default. The Shell performs the default action when it receives a null line query response (i.e., when you simply press <return>). To change the default, enter d followed by "yes", "no", or "none". The initial default is "none", which means that the command ignores null line responses, and requires explict yes or no responses.</return>

Chapter 4 describes each Shell command in detail. Those commands that use the command line parser refer you to this section for information on the standard options to avoid repetition in the text.





FUNCTIONAL COMMAND INDEX

MANIPULATING FILES/DIRECTORIES

Creating Files/Directories

CRD (CREATE_DIRECTORY) -- Create a directory
CRF (CREATE_FILE) -- Create a file
CRL (CREATE_LINK) -- Create a link
CRRGY (CREATE_REGISTRY) -- Create or modify network registry

Cataloging Files

CTOB (CATALOG_OBJECT) -- Catalog an object

UCTOB (UNCATALOG_OBJECT) -- Uncatalog a pathname without deleting the object

LD (LIST_DIRECTORY) -- List contents of a directory

ND (NAMING_DIRECTORY) -- Set or display naming directory

WD (WORKING_DIRECTORY) -- Set or display working directory

Changing File/Directory Attributes

CHN (CHANGE_NAME) -- Change an object's name
CVT_REC_UASC -- Convert files between types
OBTY (OBJECT_TYPE) -- Set or display the type of an object

Copying Files

CATF (CATENATE_FILE) -- Catenate files and write to output CPF (COPY_FILE) -- Copy a file
CPBOOT (COPY_BOOT) -- Copy system boot file
CPSCR (COPY_SCREEN) -- Copy the display to a file
CPT (COPY_TREE) -- Copy a tree
CRPAD (CREATE_PAD) -- Create a transcript pad and window
MVF (MOVE_FILE) -- Move a file
TEE -- Copy input to output and to named files

Comparing Files

CMF (COMPARE_FILE) -- Isolate differences between files CMSRF (COMPARE_SORTED_FILES) -- Isolate differences between sorted files CMT (COMPARE_TREE) -- Compare all objects in trees

Printing Files

PRF (PRINT_FILE) -- Print a file on a printer

Deleting Files/Directories

DLF (DELETE_FILE) -- Delete a file DLL (DELETE_LINK) -- Delete a link DLT (DELETE TREE) -- Delete a tree

Salvaging Files/Directories

SALD (SALVAGE_DIRECTORY) -- Salvage a directory SALVOL (SALVAGE_VOLUME) -- Verify and correct allocation of disk blocks

Protecting Files/Directories

ACL (ACCESS_CONTROL_LIST) -- List or copy an access control list EDACL (EDIT_ACCESS_CONTROL_LIST) -- Edit or list an existing ACL SALACL (SALVAGE_ACCESS_CONTROL_LIST) -- Salvage an ACL

Backing Up Files/Directories

ARCF (ARCHIVE_FILE) -- Maintain an archive file
RBAK (READ_BACKUP) -- Restore or index a magnetic tape backup file
WBAK (WRITE_BACKUP) -- Create a magnetic tape backup file

Locking/Unlocking Files

LKOB (LOCK_OBJECT) -- Lock an object
LLKOB (LIST_LOCKED_OBJECTS) -- List locked objects
ULKOB (UNLOCK_OBJECT) -- Unlock an object

Reading and Writing Files on Tape

EDMTDESC (EDIT_MAGTAPE_DESCRIPTOR) -- Create or modify magtape descriptor file RWMT (READ_WRITE_MAGTAPE) -- Read and write files on magnetic tape RBAK (READ_BACKUP) -- Restore or index a magnetic tape backup file WBAK (WRITE_BACKUP) -- Create a magnetic tape backup file

EDITING FILES

Locating Text

FPAT (FIND_PATTERN) -- Find a text pattern in a file FPATB (FIND_PATTERN_BLOCK) -- Find blocks of lines containing a pattern

Replacing Text

CHPAT (CHANGE_PATTERN) -- Replace pattern in text file
DLDUPL (DELETE_DUPLICATE_LINES) -- Strip repeated lines from a file
ED (EDIT) -- Edit a text file
EDACCT (EDIT_ACCOUNT) -- Edit registry account file

EDPPO (EDIT_PPO) -- Edit registry PPO files

EDSTR (EDIT_STREAM) -- Edit a stream

MACRO -- Expand macro definitions

OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control

TLC (TRANSLITERATE_CHARACTER) -- Replace characters

Sorting Text

CREFS (CROSS_REFERENCE_SYMBOLS) -- Cross reference symbols in file EXFLD (EXTRACT_FIELDS) -- Manipulate fields of data FLEN (FILE_LENGTH) -- Count lines, words, and characters in a file LAMF (LAMINATE_FILE) -- Laminate files REVL (REVERSE_LINES) -- Reverse each line in a text file SRF (SORT_FILE) -- Sort and/or merge text files

FORMATTING FILES

FMC (FORMAT_MULTI_COLUMN) -- Format text file into multiple columns FMT (FORMAT_TEXT) -- Format text file
OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control
PAGF (PAGINATE_FILE) -- Paginate a file to output

DEVELOPING PROGRAMS

Compiling Programs

MACRO -- Expand macro definitions

Debugging Programs

ABTSEV (ABORT_SEVERITY) -- Set or display abort severity level
DEBUG -- Invoke the Language Level Debugger
ESA (EXTERNAL_SYMBOL_ADDRESS) -- Display address of external symbol in
installed library
HPC (HISTOGRAM_PROGRAM_COUNTER) -- Make a histogram of the program counter
STCODE (STATUS_CODE) -- Translate status code value to text message
TB (TRACEBACK) -- Print traceback after a fault

Loading Programs

BIND -- Combine object modules into an executable file INLIB (INSTALL_LIBRARY) -- Install a user-supplied library. LBR (LIBRARIAN) -- Create an object module library

MANAGING YOUR NODE

CALENDAR (SET SYSTEM CALENDAR) -- Set system calendar clock. CPBOOT (COPY_BOOT) -- Copy system boot file FIND_ORPHANS -- Locate and catalog uncataloged objects.

SALVOL (SALVAGE_VOLUME) -- Verify/correct allocation of disk blocks (Appendix D) SCRTO (SCREEN_TIMEOUT) -- Set/show screen timeout TPM (TOUCH_PAD_MODE) -- Set characteristics for the touchpad

REQUESTING PROCESS/SYSTEM INFORMATION

Process

CSR (COMMAND_SEARCH_RULES) -- List or define command search rules
FST (FAULT_STATUS) -- Display fault status information
LOPSTR (LIST_OPEN_STREAMS) -- List open streams
PST (PROCESS_STATUS) -- List process internal state information
STCODE (STATUS_CODE) -- Translate status code value to text message

Node

BLDT (BUILD_TIME) -- Display time at which system was built DATE -- Display current date and time

TZ (TIME_ZONE) -- Set or display system time zone

LAS (LIST_ADDRESS_SPACE) -- List objects mapped into the address space

LLKOB (LIST_LOCKED_OBJECTS) -- List locked objects.

LVOLFS (LIST_VOLUME_FREE_SPACE) -- List free space on all logical volumes

NETSTAT (NETWORK_STATISTICS) -- Display network statistics

NETSVC (NETWORK_SERVICE) -- Set or display network services

TCTL (TERMINAL_CONTROL) -- Set or display terminal (SIO line) characteristics

Network

LCNODE (LIST_CONNECTED_NODES) -- List nodes connected to the network

LUSR (LIST_USER) -- List users logged on

LVOLFS (LIST_VOLUME_FREE_SPACE) -- List free space on all logical volumes

NETMAIN (NETWORK_MAINTENANCE) -- Control/analyze network maintenance statistics.

NETMAIN_NOTE (NETWORK_MAINTENANCE_NOTES) -- Place message in network error log

NETSTAT (NETWORK_STATISTICS) -- Display network statistics

NETSVC (NETWORK_SERVICE) -- Set or display network services

PROBENET (PROBE_NETWORK) -- Probe network and display error statistics

SETTING PROCESS CONDITIONS

Controlling Programs

ABTSEV (ABORT_SEVERITY) -- Set or display abort severity level FPPMASK (FLOATING_POINT_MASK) -- Set or display floating point error mask PPRI (PROCESS_PRIORITY) -- Set or display process priority.

SIGP (SIGNAL_PROCESS) -- Signal a process to stop RETURN -- Return from the current Shell level at a specific error value

Controlling Shells

SIGP (SIGNAL_PROCESS) -- Signal a process to stop

CRP (CREATE_A_PROCESS) -- Create a process on a remote node.

CRSUBS (CREATE_SUBSYSTEM) -- Create a protected subsystem.

CSR (COMMAND_SEARCH_RULES) -- List or define command search rules

ENSUBS (ENTER_SUBSYSTEM) -- Enter a protected subsystem at Shell command level.

RDYM (READY_MESSAGE) -- Set system ready message

SUBS (SUBSYSTEM) -- Set or display subsystem attributes

SH (SHELL) -- Invoke a Shell (command line interpreter)

SET -- Set current Shell conditions

BON -- reset SHELL -B flag

BOFF -- reset SHELL -B flag

EON -- reset SHELL -E flag

EOFF -- reset SHELL -E flag

VOFF (VERIFY_OFF) -- Reset SHELL -V flag

VON (VERIFY_ON) -- Reset SHELL -V flag

XOFF -- Reset SHELL -X flag

XON -- Reset SHELL -X flag

XSUBS (EXECUTE_SUBSYSTEM) -- Execute a Shell script protected subsystem manager LOGIN -- Log in to a running process.

WRITING YOUR OWN COMMANDS (SHELL SCRIPTS)

ARGS (ARGUMENTS) -- Echo command line arguments

EQS (EQUALS) -- Compare strings for equality

EXISTF -- Check to see if an object exists

EXIT -- Exit a loop

NEXT -- Return to the top of a loop

RETURN -- Return from current Shell level

IF -- Execute a conditional statement

FOR -- Execute a FOR loop.

SELECT -- Execute a SELECT condition

WHILE -- Execute a WHILE loop

XDMC (EXECUTE_DM_COMMAND) -- Execute a Display Manager command

SOURCE -- Execute a Shell script at the current Shell level

USING SHELL VARIABLES

DLVAR (DELETE_VARIABLE) -- Delete a Shell variable

EXISTVAR (EXIST VARIABLE) -- Check to see if a variable exists

EXPORT -- Change a Shell variable into an Environment variable

LVAR (LIST_VARIABLES) -- List name, type, and value of current variables

READ -- Set variables equal to input values

READC -- Set variables equal to input character values

READLN -- Set a variable equal to an input value

MANAGING NETWORK FUNCTIONS

Manipulating the Network Registry

CRRGY (CREATE_REGISTRY) -- Create or modify network registry
EDACCT (EDIT_ACCOUNT) -- Edit registry account file
EDPPO (EDIT_PPO) -- Edit registry PPO files
LRGY (LIST_REGISTRY) -- List contents of registry files
SALRGY (SALVAGE_REGISTY) -- Salvage network and local registries

Controlling Peripheral Devices

EDMTDESC (EDIT_MAGTAPE_DESCRIPTOR) -- Create or modify magtape descriptor files.

NETSVC (NETWORK_SERVICE) -- Set or display network services

PRSVR (PRINT_SERVER) -- Start the Print Server

Controlling Logical Volumes

CTNODE (CATALOG_NODE) -- Catalog a node in the network
UCTNODE (UNCATALOG_NODE) -- Uncatalog a node
INVOL (INITIALIZE_VOLUME) -- Initialize a disk volume (Appendix C)
MTVOL (MOUNT_VOLUME) -- Mount a logical volume
DMTVOL (DISMOUNT_VOLUME) -- Dismount a logical volume

USING MISC. UTILITIES

CALENDAR -- Set hardware clock and calendar (Appendix A)

CRUCR (CREATE_USER_CHANGE_REQUEST) -- Create a User Change Request form

DCALC (DESK_CALCULATOR) -- Evaluate logical and arithmetic expressions

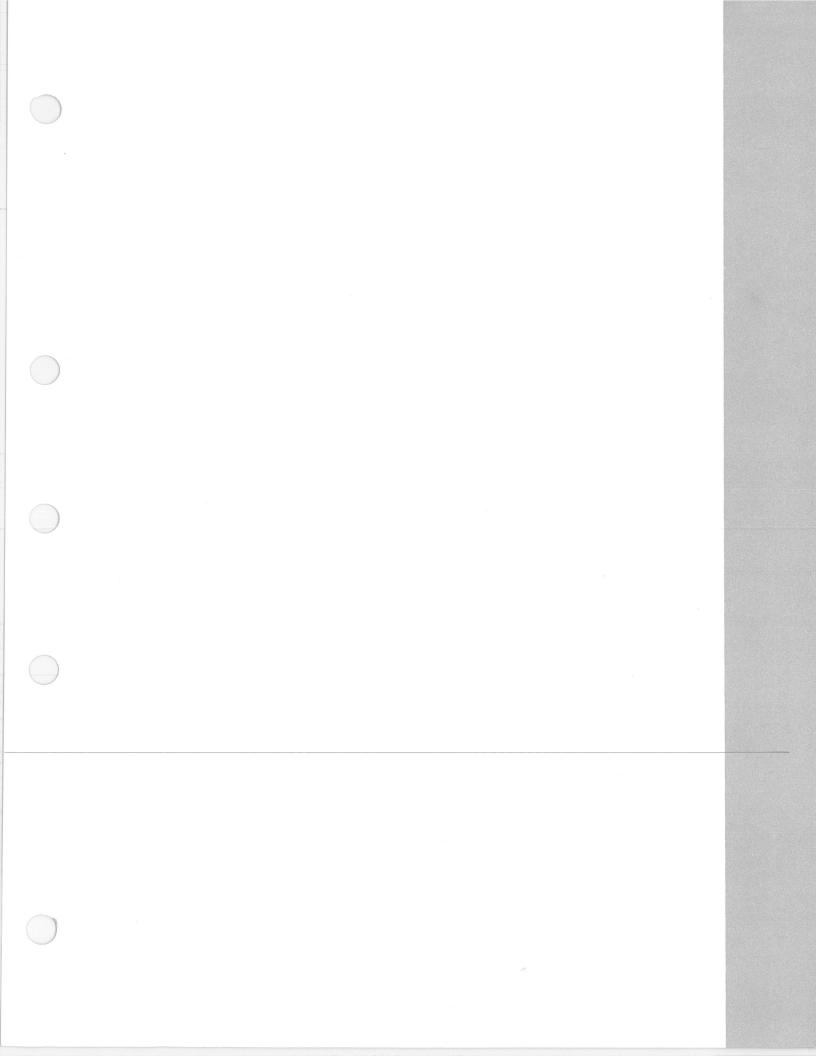
EDFONT (EDIT_FONT) -- Edit a character font (Appendix B)

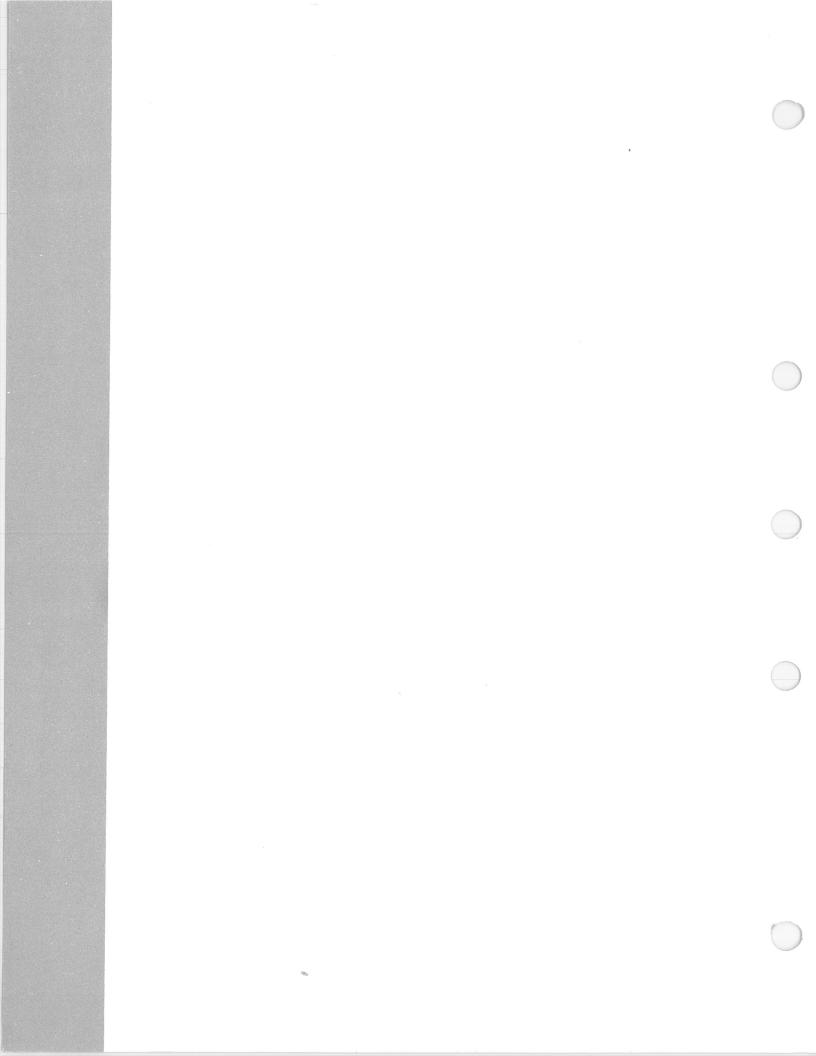
FSERR (FIND_SPELLING_ERRORS) -- Find spelling errors in a text file

HELP -- Invoke HELP facility

COMMUNICATING WITH REMOTE COMPUTERS

EM3270 -- Emulate an IBM 3270 terminal
VT100 -- VT100 terminal emulator.
VCTL (VT100_CONTROL) -- Set/display VT100 terminal characteristics.
EMT (EMULATE_TERMINAL) -- Emulate a dumb terminal
TCTL (TERMINAL_CONTROL) -- Set or display terminal (SIO line) characteristics
SIORF (SIO_RECEIVE_FILE) -- Receive a file from a remote host
SIOTF (SIO_TRANSMIT_FILE) -- Transmit a file to a remote host
SIOLOGIN (SIO_LINE_LOGIN) -- Invoke login sequence on an SIO line.
SIOMONIT (SIO_PROCESS_MONITOR) -- Support repeated logins on an SIO line.





Chapter 4 Shell Commands

ABTSEV (ABORT_SEVERITY) -- Set or display the abort severity level.

FORMAT

ABTSEV [options]

Every Shell command or program returns a completion status message to its caller. The message may indicate that the program completed successfully, or it may inform its caller of a fatal internal error. Completion status messages vary in their severity. The following completion status messages appear in order of their severity:

- 1. OK -- the program completed successfully and performed the requested action.
- 2. TRUE -- the program completed successfully; its purpose was to test a condition, and the value of that condition was TRUE.
- 3. FALSE -- the program completed successfully; its purpose was to test a condition, and the value of that condition was FALSE.
- 4. WARNING -- the program completed successfully and performed the requested action. However, an unusual (but nonfatal) condition was detected.
- 5. ERROR -- the program could not perform the requested action because of an error in the input. The output, however, is sound.
- 6. OUTPUT INVALID -- the program could not perform the requested action because of a syntactic error in the input, and the output is not structurally sound.
- 7. INTERNAL FATAL -- the program detected an internal fatal error and stopped. The state of the output is unknown.

The ABTSEV command lets you set the severity level at which a Shell command or program aborts. If a Shell command or program meets or exceeds the abort-severity level, then it (and all of its ancestors) abort.

ABTSEV works on a per Shell program basis. A new Shell or Shell program inherits its creator's abort-severity level, and the operating system restores that abort-severity level when you exit from the Shell or when the Shell program stops.

ABTSEV is an internal Shell command.

See the PGM_\$SET_SEVERITY description in the DOMAIN System Call Reference for further information on severity levels.

OPTIONS

Specifying ABTSEV without options displays the current abort severity level. All options must be specified in UPPERCASE letters.

-F[ALSE]

Set level to false.

ABTSEV (ABORT_SEVERITY_LEVEL)

-W[ARNING]

Set level to warning.

-E[RROR]

Set level to error.

-O[UTINV]

Set level to output invalid.

-I[NTFATAL]

Set level to internal fatal error.

-P[GMFLT]

Set level to program fatal error.

-M[AX_SEVERITY]

Set level to maximum severity error.

EXAMPLES

\$ abtsev
error
\$ abtsev -

\$ abtsev -W

\$ abtsev warning

Show initial setting.

Set level to WARNING.

Show new level.

ACL (ACCESS_CONTROL_LIST) -- List or copy an ACL.

FORMAT

ACL [target_object [source_object]] [options]

Every directory and file has an associated access control list (ACL) which lists users and their rights to the object. ACL lets you copy an ACL from one object to another, or display an ACL. For a detailed discussion of ACL structure and usage, please refer to the EDACL (EDIT_ACL) command description.

In addition to its own ACL, each directory contains within it two additional ACLs (called "initial ACLs"): one for new files and another for new subdirectories that are created within that directory. When you create a new file or directory, the system assigns the appropriate initial ACL stored in the parent directory. Also, when you copy files or directories to new locations in the file hierarchy, the destination object receives the appropriate initial ACL from its new parent directory. To change or display these initial ACLs stored within a directory (so that newly created objects receive new initial ACLs), use the -I, -ID, or -IF options.

The ACL command only displays ACLs or copies ACLs from one object to another. To make changes to an existing ACL, use the EDACL command.

ARGUMENTS

target_object
(optional)

Specify the object whose ACL you want to set or display. You may use a wildcard to specify this argument. DO NOT, HOWEVER, DO \$ acl /... (anything) AS THIS MAY RENDER YOUR NODE UNUSABLE. This wildcard sequence includes files in the /SYS tree, which require special ACL settings in order for system software to run.

Default if omitted: use current working directory

source_object
(optional)

Specify the file or directory whose ACL(s) is to be used to set the ACL(s) of the target object(s).

Default if omitted: display target_object's ACL

OPTIONS

The following options confine the ACL command's operation to target objects of the given type.

-D

Set or display ACLs of only those target objects that are directories. If used with -I, -ID, or -IF options, set or display initial ACLs for subdirectories.

-F

Set or display ACLs of only those target objects that are files.

The following options control the ACL command's effect on target objects. If the target object is a directory, they cause ACL to operate only on the initial ACLs stored within that directory for use on newly created objects, and not on the ACL of the directory itself. Note that this does NOT imply that all the target object(s) are directories, however. (That is what -D specifies.)

-I

Set or display initial ACLs. If you are setting the ACLs of a target directory, the source object's type (file or directory) determines which initial ACL (the one for files or the one for directories) of the target directory is set. If the target object is a file (or if a wildcarded target list includes files) and the source is a directory, you will get an error unless you have also specified -IS (so that the initial file ACL in the source directory, rather than the ACL of the directory itself, can be copied to target files). If both source and target are files, then the source file's ACL is applied to the target file, as you would expect. You must run SALD (SALVAGE_DIRECTORY) on target directories that have never contained initial ACLs (i.e., those directories created using software prior to SR4.1).

-ID

Set or display only the initial ACLs inside those target objects that are directories that apply to new subdirectories created in those directories.

-IF

Set or display only the initial ACLs inside those target objects that are directories that apply to new files created in those directories.

(Specifying both -ID and -IF is the same as -I. Neither implies -D.)

The following option specifies that one (or both) of the initial ACLs inside the source object is to be copied to the target, rather than the ACL of the source itself. This assumes that the source object is a directory and not a file, since files cannot contain initial ACLs for subordinate objects.

-IS

Copy the initial ACL(s) in the source object (which must be a directory) to the target. If there is a single target object (either a file or a directory), then the appropriate initial ACL inside the source is applied to the target. If the -I option is also specified, then both initial ACLs in the source are copied to the initial ACLs inside those target objects that are directories.

The following option specifies that all the ACLs of the target object(s) are to be set or displayed.

-ALL

Set or display all ACLs of the target object(s). If you are using wildcards to specify the target, you may qualify this action by also specifying -D or -F. If the source object is a directory, then all of its ACLs (both its own and the two initial ACLs that it applies to newly created subordinate objects) are used to set the corresponding ACLs of the target object(s). If -IS is also specified, however, the ACL of the source object itself will not be used, although all three ACLs of the target directories are still set. Thus using -ALL (with or without -IS) may be used to propagate new ACLs throughout subtrees.

The following options perform miscellaneous tasks:

-LINKS

If target_object is a wildcard that specifies link(s), operate on the link(s). By default ACL does not operate on links specified with wildcards. ACL always, however, operates on links you specify explicitly (without wildcards).

-L

List object names as the command sets ACLs.

-BR

Display ACLs only, not object names.

ACL uses the command line parser, and so also accepts the standard command options listed in Chapter 3 with the exception of the use of hyphen (-) to read data from standard input.

EXAMPLES

1. \$ acl new_file old_file Assign old_file's ACL to new_file.

2. \$ acl joe mary -i -is Set the initial ACLs inside JOE using the initial ACLs inside MARY (which must be a directory).

3. \$ acl abc?* file1 -d -if

Set the initial file ACL in all subdirectories of the current working directory whose names begin with ABC to the ACL of FILE1.

4. \$ acl abc?* dir2 -f -is Set the ACLs of all files in the current working directory whose names begin with ABC to the initial file ACL inside DIR2.

5. \$ acl abc?* dir2 -i -is

The initial ACLs in all subdirectories of the current working directory whose names begin with ABC are set using the initial ACLs in DIR2, and the ACLs of all files whose names begin with ABC are set using the initial file ACL in DIR2.

(Adding -D would confine the operation

to directories.)

6. \$ acl abc?* dir2 -all

The ACLs of all files matched are set using the initial file ACL in DIR2. The ACLs of all directories matched are set using the ACL of DIR2 itself. The initial ACLs inside those matched directories are set using the initial ACLs inside DIR2.

7. \$ acl abc?* dir2 -all -is

The ACLs of all files matched are set using the initial file ACL in DIR2. The ACLs of all directories matched are set using the initial directory ACL in DIR2. The initial ACLs inside those matched directories are set using the initial ACLs inside DIR2.

ARCF (ARCHIVE_FILE) -- Maintain an archive file.

FORMAT

ARCF command arcname [pathname ...]

ARCF collects sets of files into one large file and maintains that file as an archive. Files can be extracted from the archive, new ones can be added, old ones can be deleted or replaced by updated versions, and data about the contents can be listed. Only text files can be archived.

Files to be added to an archive must exist as files with the name given. Files that are extracted from an archive will be written to files with the name given. Files that are added to archives can, of course, be archive files themselves. Any number of files can be nested this way. Thus, ARCF can be used to maintain tree-structured file directories.

NOTE: When you use the update and print commands, the files are updated and printed in the order they appear in the archived file, not in the order listed on the command line.

ARGUMENTS

command
(required)

Specify the operation to perform on the archive file arcname. Follow the command with V to get verbose output. Possible commands are:

-D	Delete the named files from the archive. If the
	V option is used, filenames are displayed on
	the standard output as they are deleted from
	the archive

-P Write the named files on standard output.

The V option causes the filenames to precede the file.

-T Write a table of contents for the archive file.

Normally, the table contains only the filename. If the V option is used, the table also includes the file's length, type, and date and time of the last change.

-U Update the named archive by replacing existing files, or adding new ones at the end. If no filenames are given, all possible files in the archive will be updated with files of the same name in the current directory. If the archive file does not exist, it will be created with the name given. If the V option is used, filenames are displayed on standard output as files are written to the new archived file.

ARCF (ARCHIVE_FILE)

-X

Extract the named files from archive. Write each to a file with the same name. If the file already exists, the new version replaces the old. If the V option is added, filenames are displayed on standard output as files are extracted.

V

Request verbose output. This command can follow any of the other commands (see example below), and will cause the archiver to print additional information, generally filenames, on standard output. Its specific action for each command has already been described.

Arcname (required)

Specify name of archive file being created or maintained.

pathname (optional)

Specify name of file to be added or deleted from the archive. Multiple names are permitted, separated by blanks. Specifying a hyphen as a filename will cause further names to be read from standard input, one per line.

Default if omitted: perform action on all files in the archive (except -D, which requires that names be explicitly given).

EXAMPLES

1. \$ arcf -uv my_archive stamps
 stamps
\$

Update archive file "my_archive" with a new copy of the file "stamps", returning verbose output.

2. \$ arcf -tv my_archive stamps 330 local 02/18/83 13:53:07

Report on the contents of the archive.

ARGS (ARGUMENTS) -- Echo command line arguments.

FORMAT

ARGS [-ERR[OUT]] string ...

ARGS writes its arguments, one per line, to standard output unless -ERR is specified. Use it to write to files by redirecting standard output into a file with the ">pathname" expression. The ARGS command is useful for inserting messages and diagnostics to be reported to the display into Shell scripts and for inserting lines of text into files.

ARGUMENTS

string

(required)

Specify the string of characters to be written. Multiple strings are permitted; separate strings with blanks. Strings are written one per line. To write phrases containing literal blanks, enclose strings in quotes.

OPTIONS

-ERR[OUT]

Write the string(s) to error output instead of standard output. This option is useful for writing to the transcript pad (where error output is usually directed) from an ARGS command inside a pipeline, since standard output is then connected to the pipe.

EXAMPLES

- 1. \$ args Hi there
 Hi
 there
 \$
- 2. \$ args "Hi there" "Mary"
 Hi there
 Mary
 \$
- 3. \$ args "Hi there, Mary." >my_file Write "Hi there, Mary." into the file MY_FILE in the current working directory.

BIND -- Combine object modules into an executable file.

FORMAT

BIND [pathname] [option] ...] [-]

The binder combines two or more object modules into one executable object module. It resolves all references to global symbols and combines sections that have the same name. As input, it accepts a list of object module files; as output, it produces one object module file and an optional map file.

For more detailed information on the actions of the binder in association with the available language processors, refer to the DOMAIN FORTRAN Language Reference, the DOMAIN Pascal Language Reference, the DOMAIN C Language Reference, and the DOMAIN Lisp Language Reference. A full description of the binder and its associated Librarian utility is available in the DOMAIN Binder and Librarian Reference.

Prompting

The optional hyphen at the end of the command line requests the binder to prompt for additional arguments on subsequent lines. The hyphen is valid only on the line containing the BIND command, and must be the last item on the line.

If you request prompting, the binder processes the arguments on the command line, then displays an asterisk (*) on standard output. The binder then reads further arguments from the asterisk prompts. For example:

```
$BIND peter.bin - <RETURN>
*paul.bin -ALLMARK -B name.bin <RETURN>
*time.bin -UNMARK date -UNMARK year <RETURN>
*john.bin -map <RETURN>
*
```

Prompting ends when you enter the -END switch or press <RETURN> in response to the asterisk. After prompting ends, the binder finishes processing the input files and creating the output files.

Comments

During a prompting session or in a Shell script, you can include comments among the arguments. Comments must be delimited by braces, as in this example:

```
$BIND
*london.bin
*{a map of Paris might be useful}
*paris.bin -map
*-END
```

The binder ignores the comment.

Options

The binder processes the arguments and options sequentially. Most options apply to files specified later in the command string, but have no effect on the files previously specified. In this manner, some options can be turned on and off from one file to the next.

Binder Errors

If a problem occurs during binding, the binder displays a message on error output. The message indicates the nature and severity of the problem. Error-level messages are issued for fatal conditions, which prevent the binder from producing an output file. Warning-level messages indicate conditions that do not prevent the binder from producing an output file, but may mean that the file's contents are not what you might expect.

Exiting

To exit from the binder without finishing, you can type -QUIT. In response to -QUIT, the binder stops immediately and closes all input and output files. Because processing is not complete, the output files may not contain meaningful data.

ARGUMENTS

BIND specified without arguments will cause prompting for required information.

pathname ... (optional)

Specify one or more input files to be bound into the object module. Multiple pathnames and wildcarding are permitted. If a pathname refers to a library file, no modules from the library are included in the bound object module unless:

- they are implicitly referenced by some other module which is included in the bound object module, or
- they are explicitly included with the -INCLUDE option (described below).

Default if omitted: BIND prompts for pathname.

OPTIONS

Default options are indicated by "(D)."

-ALIGN section-name [LONG | QUAD | PAGE]

Cause the named section to be aligned, at load time, on either a 32-bit (LONG), a 64-bit (QUAD), or an 8192-bit (PAGE) boundary. By default, all sections are aligned on a LONG boundary.

-ALLRES[OLVED]

Cause the binder to exit in an error state if unresolved globals remain at the end of the bind step. This is useful for control of shell scripts where an error exit of the binder is desired for this condition. Note that this option is sensitive to the -SYS[TEM]

option. If -SYSTEM is used, then ANY unresolved global remaining at the end of the bind step will cause the error exit. If -SYSTEM is NOT used, then ONLY those unresolved globals which could not be resolved via the current known global table (KGT) of the process running the binder will cause the error exit.

-B[INARY] pathname

Write the output object module to the named file.

-BDIR pathname

Add a pathname to the search hierarchy of directories for include file names. The hierarchy applies only to file name strings which do NOT begin with '.', '~', or '/'. The binder first tries to open the include file name as given. Failing that, it prepends the -BDIR pathnames to the given file name in the same order as supplied on the command line.

-END

End of commands (same as blank line).

-EXACTCASE

Set binder to be case-sensitive on all names. This is most useful when binding C object modules, which can output case-sensitive names.

-GLO[BALS]

Write currently defined global symbols to error output.

-H[ELP]

Print this list of commands.

-LOCALSEARCH

Search each library completely and attempt to satisfy all unresolved globals before searching the next library.

-MAK[ERS]

Display information about the compiler(s) and binder used to create the object module specified by 'pathname'. The output will appear in the transcript pad (and in the map output if -MAP is also specified).

-MAP

Write complete map to standard output.

-MOD[ULE] name

Name the output object module. The default name is the first input module name.

-NOEXACTCASE

(D)

Set binder to ignore case differences on names.

-NOLOCALSEARCH

(D)

Search each library once to resolve undefined globals.

-NOUND[EFINED]

Suppress report of undefinded globals.

-Q[UIT]

Exit from binder without finishing.

-READONLY[SECTION] name

Change named section from read/write to read-only. This option changes sections such as FORTRAN COMMON blocks from read/write to read-only to improve program invocation time. Only those sections containing tables initialized to certain values by FORTRAN DATA statements (or the like), and that never change during program execution, are eligible.

-SEC[TIONS]

Write current section definitions to error output.

-SET_VER[SION] nn.mm

Set program revision number to the number specified. 'nn' and 'mm' are integers (i.e., '-SET_VERSION 1.23').

-SORTL[OCATION]

List defined globals, sorted by global location (section number and offset).

-SORTN[AMES]

(D)

List defined globals sorted by global name.

-SYS[TEM]

Make system global symbols visible.

-SYSTYPE type

Override all system information from within the object modules, and force the output object module to have the specified system type marker. This is intended primarily for use with DOMAIN/IX. Valid 'type' specifiers are "sys3", "sys5", "bsd4.1", "bsd4.2", and "any". The -SYSTYPE option overrides all system information from within the object modules.

-UND[EFINED]

Write currently undefined global symbols to error output.

- (hyphen alone)

Request binder prompting for further arguments. Use only on the line containing the BIND command. The hyphen must be the last item on the line.

The following options are for use with user-installed libraries.

-ALLMARK

Begin marking all defined global symbols.

-ALLUNMARK

(D)

Stop marking defined global symbols.

-INCLUDE {module-name | -ALL}

Include a specific module from a library, or all the modules from a library, in the bound object module. This option must follow a pathname which refers to a library file or an error message is generated. Note that this option is the only way to include a module from a library in a bound object module, unless the library module is explicitly referenced by a module which is already in the bound object module.

-LOOKS[ECTION] name

Set "look at installed sections" attribute on named data section.

-LOOKS[ECTION] -ALL

Set "look at installed sections" attribute on all subsequent data sections.

-NOLOOKS[ECTION] name

Remove "look at installed sections" attribute from named data section.

-NOLOOKS -ALL (D)

Stop setting "look at installed sections" attribute on all data sections.

-MARK global_name

Mark the named global symbol.

-MARK -ALL

Same as -ALLMARK.

-MARKS[ECTION] name

Mark named data section for installing.

-MARKS[ECTION] -ALL

Mark all subsequent data sections for installing.

-UNMARK global_name

Remove mark from the named global symbol.

-UNMARK -ALL

Same as -ALLUNMARK.

-UNMARKS[ECTION] name

Remove installing mark from named data section.

-UNMARKS -ALL (D

Stop marking all data sections for installing.

-MULTIRES

Report errors if multiple resolutions in object module libraries

exist.

-NOMULTIRES

(D) Suppress error reporting for mulitple resolutions in object

module libraries.

-XREF Send a binary cross-reference to standard output. This will

cross-reference only files following this option in the command

line.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

Turning Options On and Off

Because the binder processes arguments and options sequentially, most options apply to files specified later in the command string, but have no effect on the files previously specified. In this manner, some options can be turned on and off from one file to the next.

EXAMPLES

- 1 \$BIND <RETURN> *paul.bin -ALLMARK -B name.bin <RETURN> *time.bin -UNMARK date -UNMARK year <RETURN> arguments in response to *john.bin -map <RETURN> *<RETURN> >
- 2 \$BIND *london.bin *{a map of Paris might be useful} *paris.bin -map *-END

BIND specified alone on command line; enter asterisk. Press <RETURN> to end prompting.

To include comments among arguments, enclose the comments in braces.

BLDT (BUILD_TIME) -- Display time at which operating system was built.

FORMAT

BLDT [pathname] [option]

BLDT displays the time at which the running version of AEGIS was built.

ARGUMENTS

pathname

(optional)

Display the build time of the node whose network root directory is "pathname."

Default if omitted: display build time of current node

OPTIONS

-N node_id ...

Display build time of specified node[s]. The node_id value may be an entry directory (i.e., //FROTUS) or a hex node ID (i.e., 27A3).

-A

Display build time of all nodes.

EXAMPLES

- 1. \$ bldt
 **** Node 532 **** "//paris"
 AEGIS, revision 5.0, built on Friday, June 5, 1982 2:29:44 am (EST).
- 2. \$ bldt //os **** Node 21 **** "//london" AEGIS1, revision 6.0x, built on Tuesday, June 24, 1983 9:01:00 pm (EDT).
- 3. \$ bldt -n 74 **** Node 74 **** "//munich" AEGIS1, revision 6.0x, built on Monday, July 1, 1983 10:26:41 am (EDT).

BOFF -- Deactivate the Shell's -B flag.

FORMAT

BOFF

BOFF turns off the Shell's -B (display output of background process) flag, which is turned on by the BON command or the -B option on the SH command line. When the flag is off, the output of background processes created with the & parsing operator is sent to /DEV/NULL. This background process output is sent to /DEV/NULL by default.

BOFF requires no arguments or options.

BON -- Activate the Shell's -B flag.

FORMAT

BON

BON activates the Shell's -B (display output of background process) flag. The flag can also be activated by using the -B option on the SH command when the Shell is invoked. The BOFF command deactivates the -B flag. By default, the flag is off when a Shell is invoked.

This flag causes the Shell to send the output of a background process (created with the "&" parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it was invoked.

If BON is turned on in a Shell script, it remains on until that Shell script exits, or until it is over-ridden by a BOFF command in a nested Shell script. When a Shell script exits, the state of execution tracing is returned to the state in effect just before the script was invoked.

BON requires no arguments or options.

CALENDAR (SET SYSTEM CALENDAR) -- Set system calendar clock.

FORMAT

CALENDAR (from the Shell)
EX CALENDAR (from the Mnemonic Debugger)

The calendar utility is used to set or reset the calendar clock in a node. It can also be used to update the last valid time known to the system, which is stored on the boot volume. Normally, the clock is set at the factory, and there is no need to reset it. Care must also be taken if setting the clock backwards in time, since duplicate UIDs may be generated, resulting in the loss of files. For information on changing the timezone, see the TZ (TIMEZONE) command description.

Please refer to the appropriate appendix for complete information on the use of this software tool.

CALENDAR prompts for all required arguments and options.

CATF (CATENATE_FILE) -- Read file(s) and write to standard output.

FORMAT

CATF [pathname ...]

CATF reads input files in order and writes them to standard output.

ARGUMENTS

pathname

(optional)

Specify file(s) to write to standard output. If multiple pathnames are given, they are read and written in the order that they appear on the command line.

Default if omitted: read standard input

OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

- 1. \$ catf garbage Writes the file "garbage" on standard output.
- 2. \$ catf garbage trash >collector

Concatenates the file "garbage," the lines read from standard input, and the file "trash," and writes the result in the file "collector."

3. \$ catf collector >>junk

Appends the contents of "collector" to the file "junk."

CHHDIR (CHANGE_HOME_DIRECTORY) -- Change a login home directory.

FORMAT

CHHDIR pathname

The login home directory contains your initial working and naming directories. After login, you are automatically in your login home directory. Use CHHDIR to change your login home directory.

In order for this command to work properly, the network registry must have been sealed so it is owned by the LOGIN subsystem. See — the command descriptions for CRSUBS, SUBS, and ENSUBS for more information on protected subsystems and the objects they protect.

ARGUMENTS

pathname

(required)

Specify name of new login home directory.

EXAMPLES

\$ chhdir //user/john

Set new login home directory to //user/john.

CHN (CHANGE_NAME) -- Change an object's name.

FORMAT

CHN old_name [new_name] [old_name [new_name] ...] [options]

CHN changes the name of a file, directory, or link. CHN works with the rightmost component ("leafname") of the old name (see EXAMPLES).

This command cannot be used to change the name of a directory embedded in a complete pathname, which would result in the file's relocation to some other part of the naming tree. For instance,

\$ chn //et/mary/letters //et/fred/letters

is illegal. Use the MVF (MOVE_FILE) command for that operation.

ARGUMENTS

Multiple 'old_name'/'new_name' pairs and pathname wildcarding are permitted.

old_	name
(requ	ired)

Specify the current pathname of the object to be renamed.

new_name

(optional)

Specify the new name of the object. The new name may be derived from the old name. 'New_name' may be omitted entirely if -D, -Y, or -U are specified. Otherwise, some portion of it is required. Names may be 1 to 32 characters long.

Default if omitted: derive 'new_name' from 'old_name'.

OPTIONS

-D Append today's date (month and day) to 'new_name' in the form "new_name.mm.dd"

Append today's date (year, month, and day) to 'new_name' in the form "new_name.yy.mm.dd"

-U Force 'new_name' to be unique by appending a sequence number to the end of the name until it becomes unique.

-S List names changed on standard output.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1.	\$ chn fritz henri	Change the name "fritz" to "henri" in the current working directory.
2.	\$ chn henri mike peter paul	Change henri to mike and peter to paul.
3.	\$ chn (a b c) = zorp	Change a b and c to a.zorp, b.zorp and c.zorp.
4.	<pre>\$ chn /my/stuff/lips red_lips</pre>	Change the file "lips" to "red_lips" in the directory "/my/stuff."
5.	\$ chn henri -d henri.07.19	Change henri to henri mm dd where mm is the current month (01-12) and dd is the current date (01-31).
6.	\$ chn joe -u joe.1	Change joe by appending sequence number to end of file name.

CHPASS (CHANGE_PASSWORD) -- Change a login password

FORMAT

CHPASS [new_password]

CHPASS changes your login password to 'new_password.' CHPASS allows you to change your password from the Shell command level.

In order for this command to work properly, the network registry must have been sealed so it is owned by the LOGIN subsystem. Refer to CRSUBS, SUBS, and ENSUBS, plus HELP PROTECTION, for more information on protected subsystems and the objects they protect.

Your login password can also be changed using the -P option under the L (LOGIN) Display Manager command.

ARGUMENTS

new_password (optional)

Specify new password. Omitting this option causes CHPASS to prompt for your new password. Input echo is disabled. A second prompt verifies the password and guards against typing mistakes.

Default if omitted: prompt up to three times for password.

EXAMPLES

\$ chpass sesame Set new login password to "sesame."

CHPAT (CHANGE_PATTERN) -- Replace pattern in text file.

FORMAT

CHPAT [options] [pathname ... -P] [from pattern | pat ...] from pattern [to_expression]

CHPAT copies every line from its input files to its output files, globally substituting the text replacement pattern "to expr" for each occurrence of "fr pat" in those lines which match according to the "pat" arguments and the option flags.

Refer to the descriptions of the ED (EDIT), FPAT (FIND PATTERN), and EDSTR (EDIT STREAM) commands for related information.

ARGUMENTS

pathname

(optional)

Specify name of file to be searched. Multiple pathnames and wildcarding are permitted. If this argument is present, it must be followed by "-P" to separate the pathname(s) from the search patterns on the command line.

Default if omitted: search standard input.

from pattern (required)

Specify target text string (a regular expression) for substitution or deletion. If the string includes the characters % \$ [] { }!* or any other Shell special characters, enclose it in quotes to avoid unpredictable results.

to_expression (optional)

Specify replacement string. If no replacement is specified, the 'from pattern' string will be deleted. If regular expressions defining a range of text ('pat' argument) are present, you must use a literally null 'to expression' ("") to delete 'from pattern'.

pat

(optional)

Specify range of text for which the substitution is to apply, in the form of a regular expression. Multiple expressions separated by blanks are permitted. Unless modified by options, any line of text matching any pattern is replaced and all lines, changed or not, are written to output.

Default if omitted: use 'from_pattern' to select matching lines.

OPTIONS

-A

Select only lines that match all of the leading expressions, in any order.

CHPAT (CHANGE_PATTERN)

-X Select only lines that match none of the leading expressions.

-O Write only the selected lines to standard output. By default,

CHPAT writes all lines to output.

-L List name(s) of input file(s) on output file(s) as the input file(s)

are searched.

-OUT pathname

Specify name of output file. Pathname may be derived from the input file name. If this option is omitted, output is written to standard output.

EXAMPLES

1. \$ chpat foo bar

Changes all occurrences of 'foo' in standard input to 'bar' and writes the results to standard output.

2. \$ chpat '%This' " *" ' '

In lines starting with 'This', it changes all occurrences of multiple spaces to a single space.

3. \$ chpat '%This' '%That' " *" ' '

Like 2., but works on lines starting with either 'This' or 'That'.

4. \$ chpat -a '%when' 'only\$' ';' ':'

In lines that start with 'when' and end with 'only', change all semicolons to colons.

5. \$ chpat -x not none some all

In lines that do not contain either 'not' or 'none', change all instances of 'some' to 'all'.

6. \$ chpat erase

Delete (replace with nothing) all occurences of 'erase'. Exactly the same effect can be obtained with:

\$ chpat erase ''

7. \$ chpat -o other_opts pat... fr_pat to_pat

Is exactly the same as

\$ fpat pat... | chpat other opts fr pat to pat

Change all occurrences of the string "if x=y" to "if (x=y)" in all Pascal source files (files ending with '.pas') and put the output for 'X.pas' in 'X.pas.new'

CMF (COMPARE_FILE) -- Identify differences among files.

FORMAT

CMF file_a [...file_e] [options]

CMF compares the contents of two to five ASCII files and reports the differences on standard output. This command works only on files: to compare directory trees, use CMT (COMPARE_TREE).

ARGUMENTS

file_a

(required)

Specify pathname of original file; all differences are reported in relation to this file. Wildcarding of this pathname is permitted to achieve multiple comparisons.

file_b ... file_e

(optional)

Specify descendants of 'file_a'. If more than one file is specified, you may use a hyphen (-) to cause standard input to be read in place of a pathname. Pathnames may be derived from 'file_a'.

Default if omitted: read standard input

OPTIONS

-R pathname

Report all differences to the specified report file, in addition to reporting to standard output. This pathname may be derived from 'file_a' (if 'file_a' is wildcarded) to produce one report for each comparison. If 'file_a' is wildcarded and this report file name is NOT derived, then all reports are concatenated into the single report file.

-TB

Include trailing blanks in the comparison. By default, ignore trailing blanks.

-BR

Display only line numbers of lines containing discrepancies. By default, display both line numbers and line contents.

-L

Display names of files being compared before each comparison is performed. This is useful when wildcarded pathnames are specified.

-M n

Set the minimum number of lines for a rematch to 'n.' This is the minimum number of lines, following a reported difference, which must match for CMF to consider the files synchronized. The default value is 3.

EXAMPLES

Assume that file "file1" contains

Fourscore and seven years ago, our fathers brought forth

and "file2" contains

Eighty-seven years ago, our fathers brought

CMF produces the following output when "file1" is compared to "file2."

\$ cmf file1 file2

A1 Fourscore
A2 and seven
changed to
B1 Eighty-seven

A5 forth deleted before end of file B

2 discrepancies found.

CMSRF (COMPARE_SORTED_FILE) -- Find lines common to two files.

FORMAT

CMSRF [options] file1 [file2]

CMSRF reads sorted files, 'file1' and 'file2', and produces 1-, 2-, or 3-column output. Column 1 contains lines found only in 'file1', column 2 contains lines found only in 'file2', and column 3 contains lines found in both files. The number option, -N, specifies which columns you want to print. To compare unsorted files, use CMF (COMPARE_FILE).

ARGUMENTS

Use of a hyphen for either file name will cause the data to be read from standard input.

file1

(required)

Specify first file for comparison.

file2

(optional)

Specify second file for comparison.

Default if omitted: compare file1 to standard input.

OPTIONS

If no options are specified, CMSRF produces a complete 3-column report.

-n

Specify number(s), where n is an integer sequence representing the following:

- 1 Report only lines exclusive to 'file1'.
- 2 Report only lines exclusive to 'file2'.
- Report only lines commmon to both files.

EXAMPLES

1. \$ cmsrf -12 //us/sorted stuff.c

Compare '//us/sorted_stuff.c to standard input and report lines found in either place, but not both.

2. \$ cmsrf -3 //us/sorted_stuff.a //us/sorted_stuff.b

Report only common lines for both files.

CMT (COMPARE_TREE) -- Compare source tree to target tree.

FORMAT

CMT source_pathname target_pathname [options]

CMT compares all the objects in the source tree against all objects in the target tree. CMT reports any objects cataloged in the source that do not also appear in the target. If the target contains objects that do not appear in the source, however, the differences are ignored.

CMT compares objects based on their internal representation, unlike CMF (COMPARE_FILE), which treats its input data as ASCII text streams and compares them as such.

Both the source and target pathnames must specify the same type of object, either a directory or a file. CMT, however, can compare objects of any type, unlike CMF, which compares only text files.

If CMT encounters differences, it reports that the objects are different and continues the comparison with the next object.

ARGUMENTS

Both the source and target pathnames must specify the same type of object, either a directory or a file. Use of wildcards in pathnames is permitted. Multiple source/target pairs are permitted.

source_pathname

(required)

Specify source tree.

target_pathname

(required)

Specify target tree. Name may be derived from 'source pathname.'

OPTIONS

If no options are specified, CMT will only report the names of directories and files with differences in source and target trees.

-L List all directories and files compared.

-LD List all directories compared.

-LF List all files compared.

-AE Abort on the first mismatch, or if the source tree contains a name not found in the target tree. By default, the comparison

continues after the mismatch is reported.

CMT (COMPARE_TREE)

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

Assume that the directories "dir1" and "dir2" each contain three files called "a," "b," and "c," and that the contents of the "b" files differ. Following is the result of a comparison of those two directories.

\$ cmt dir1 dir2
*** compare failed at file loc 0 SRC: 10002 DST: 100011
dir1/b - compare failed (from US / file utility)

CPBOOT (COPY_BOOT) -- Copy the system boot file SYSBOOT.

FORMAT

CPBOOT source directory { target directory | -DEV CT }

CPBOOT copies the system boot file SYSBOOT from one directory to another. The sysboot file is used by the bootstrap PROM to start the system. CPBOOT is useful for copying SYSBOOT to a floppy disk, thus making the stand-alone utilities (SAU) directory on the floppy disk accessible from the boot PROM. You may also use it to update a Winchester disk when a new software release is distributed.

If you wish to build a bootable cartridge tape, "-DEV CT" should be specified in place of the target directory. This will copy CTBOOT -- the cartridge tape version of SYSBOOT -- from the source directory (usually /SYS) onto the beginning of the cartridge tape. (Note: subsequent WBAKs to the tape should use the -SYSBOOT option to avoid overwritting CTBOOT on the tape.)

ARGUMENTS

source_directory

(required)

Specify directory containing the file SYSBOOT or CTBOOT.

target directory

(optional)

Specify directory to which SYSBOOT is to be copied. This

must be the entry directory on the target logical volume.

Default if omitted: must use -DEV.

OPTIONS

-DEV CT

Specify that you wish to build a bootable cartridge tape. This option must be specified if you omit the 'target directory' argument.

EXAMPLES

- \$ cpboot /flpa / Copy the SYSBOOT file from the directory "/flpa" to the current node entry directory.
- \$ cpboot /sys -dev ct Copy the CTBOOT file from the directory "/sys" to the cartridge tape.

CPF (COPY_FILE)

CPF (COPY_FILE) -- Copy a file.

FORMAT

CPF source_pathname [target_pathname] [options]

CPF copies a file from the source pathname to the target pathname. CPF only copies files; see CPT (COPY_TREE) for copying directories and their subordinate objects.

ARGUMENTS

source_pathname

(required)

Specify file to be copied. If the source pathname is a link name, CPF resolves the link and copies the file to which the link refers.

target_pathname

(optional)

Specify target for copy. If target_pathname is a directory, source_pathname is copied into this directory. Target must NOT be a link.

Default if omitted: copy 'source_pathname' into current working directory

Multiple source/target pairs and pathname wildcarding are permitted.

OPTIONS

Default options are indicated by "(D)."

-C (D) Create source file at target. An error occurs if the target file already exists.

-R Replace target with copy of source.

-LF List files copied.

-LDL List files deleted as a result of a "replace" (-R).

-CHN

Use with -C to change the name of an existing object with the target pathname before the copy is made. Use with -R to change the name of a target file if it is in use and cannot be deleted.

-DACL (D) Apply the target directory's default ACL for files copied. In addition to its own ACL, each directory has two default ACLs, one for its files and another for its subdirectories. Unless you specify -SACL (below), the system assigns to the target file its parent directory's default ACL for files.

-SACL Retain the source file's ACL.

-SUBS Retain source ACL for objects which belong to subsystems.

-F

Force deletion of target object if 'p' rights are present.

-DU

Delete when unlocked. This option is useful with -R. If the object to be replaced is locked when CPF is invoked, the replace operation will be performed when the object is unlocked.

-PDT

Preserve the source file's modification and used times.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1. \$ cpf /latest/wbak wbak.latest
\$

Copy the file "wbak" from the "/latest" directory to the current directory, and call it "wbak.latest".

2. \$ cpf /latest/com/wbak /com -r
\$

Copy the file "/latest/com/wbak" to the "/com" directory, replacing the existing "/com/wbak".

3. \$ cpf /games/space?* -lf
 (file) "spacewar" copied.
 (file) "spacebunny" copied.
 (file) "space_shot" copied.
 \$

Copy and list all files in the "/games" directory starting with "space"to the working directory.

4. \$ cpf ?*.pas backup/=.12.07

Copy all files in the working directory with the suffix ".pas" to the directory "backup", appending a date.

CPL (COPY LINK)

CPL (COPY_LINK) -- Copy a link.

FORMAT

CPL linkname [pathname] ... [options]

CPL copies a linkname to the target object.

ARGUMENTS

Multiple linkname/pathname pairs and wildcarding are permitted.

linkname

(required)

Specify the name of the link to be copied.

pathname

(optional)

Specify the target pathname of the copied link. If 'pathname' is a linkname, then this link is created or replaced (depending on various options below). If 'pathname' is a directory, then the link text is copied into this directory. In no case is the object to which the link refers affected: only the text of the link itself.

Default if omitted: copy link into current working directory.

OPTIONS

Default options are indicated by "(D)."

-C (D) Create source link at target. An error occurs if the target link already exists.

-R Replace target with copy of source.

-LL List links copied.

-LDL List links deleted because of replacement (-R).

-CHN Change name of existing link with target_pathname before copying.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1. \$ cpl //ai/sources /progs

Copy the link "//ai/sources" to the node entry directory as "progs".

2. \$ cpl //zorba/sys/print /sys -r Copy the link "/sys/print" from the node whose entry directory is "zorba" to the local /sys directory, replacing any existing link.

CPSCR (COPY_SCREEN) -- Copy the current display to a file.

FORMAT

CPSCR pathname [-INV]

CPSCR copies the current screen image (without clearing it) to the file you specify. Use the PRF (PRINT_FILE) command with the -PLOT option to print the file.

Use the Display Manager command CPO to copy the screen without creating a new process window or changing the current transcript pad. CPO invokes the CPSCR command from the Display Manager without creating a pad or window. Thus, press < CMD > then type:

CPO /COM/CPSCR pathname

You may copy small portions of the screen (such as a single window) with the DM command XI.

ARGUMENTS

pathname

(required)

Specify file to which the screen is copied.

OPTIONS

-INV

Invert image. Use this option to store the image in reverse video. Black screen pixels become white and white screen pixels become black.

EXAMPLES

- 1. \$ cpscr //us/looky_there -inv Invert and copy the current screen image to the specified file. Since the command line is echoed in the Shell's process transcript pad prior to execution, this command will appear in the resulting image.
- 2. <CMD>
 Command: cpo /com/cpscr //us/looky there -inv

Same result as in example 1, but the CPSCR line will not appear in the plotted output. CPT (COPY_TREE) -- Copy a directory tree.

FORMAT

CPT source_pathname target_pathname ... [options]

CPT is a multipurpose tool for copying, merging, and replacing files, directories, and links. To copy files only, use CPF (COPY_FILE).

ARGUMENTS

Multiple source/target pairs and wildcarding are permitted.

source_pathname

(required)

Specify the file, link, or directory tree to be copied. CPT does not change the contents or link references of the source, so errors that occur will leave the source unaffected.

target_pathname

(required)

Specify the file or directory tree to be created, replaced, or merged. The target pathname may be derived from the source pathname. The target can NOT be a link. In addition, the target can NOT be a logical volume entry directory, or the network root unless the -MD option is specified.

OPTIONS

Default options are indicated by "(D)."

-AF date

Copy only objects whose dtms (date-times) are after the given date and time: [[[yy]yy/]mm/dd][.][hh:mm[:ss]] | TODAY. The date defaults to today, and the time to midnight, if either are omitted from 'date'.

-BE date

Copy only objects whose dtms are before the given date and time: [[[yy]yy/]mm/dd][.][hh:mm[:ss]] | TODAY. The date defaults to today, and the time to midnight if either are omitted from 'date'.

-C (D)

Create source at target. If the file or directory already exists, an error will occur and processing will continue to the next source/target pair. Not valid if -MS, -MD, or -R is specified.

If the source is a file, CPT copies it to the target. If the source is a directory, CPT copies the directory to the target. It then copies every file cataloged in the directory (and all subdirectories) until it reaches the end of the tree.

Each link name in the source tree is created as a link name in the target, but the object that the link references is not copied. If 'source_pathname' is itself a link, however, the link is resolved and the object to which it points is copied to the target.

-R

Replace target with source. Not valid if -C ,-MS, or -MD is specified. CPT deletes the tree starting at the target pathname and copies the entire source tree in its place. If no target tree by the specified name exists, CPT creates one and duplicates the source.

-MS

Merge source and target if both are directories. Not valid if -C or -R is specified. If the target does not exist, CPT duplicates the source at the target. If the target exists, CPT merges the source into the target, replacing files and links, and combining directories.

If both the source and the target are directories, CPT merges their contents as described below. Otherwise, CPT deletes the target and replaces it with the source.

To merge directories, CPT compares their contents, object by object. Objects that exist in the source but not in the target are created in the target. Objects that exist in the target but not in the source remain unchanged. Files and links with the same name in both the source and the target are deleted from the target and replaced by the source version. Directories with the same name in both source and target are merged. CPT continues this process interactively until it reaches the end of the source tree.

-MD

Merge source and target if both are directories. Similar to -MS except that files and links with the same name in both source and target are left unchanged in the target.

-F

Force deletion of target object if 'p' rights are present.

-DACL (D)

Apply the target directory's default ACLs. In addition to its own ACL, each directory has two default ACLs, one for its files and another for its subdirectories. -DACL causes CPT to apply the target directory's default ACLs to each subdirectory and file it copies. The -SACL option causes each object to retain its original ACL.

-SACL

Retain the source ACL.

-CHN

Use with -C to change the name of a target before source is copied. Use if target_name already exists. Use with -R, -MS, and -MD to change the target_name if target is in use.

-SUBS

Retain the source ACLs for objects which belong to subsystems.

-PR pathname

Preserve the object 'pathname' in target when another object with the same name exists in the source. Valid with -MS option only.

-PDT

Preserve the source's modification and used times.

The following five options allow you to monitor CPT's operation. You can use -LD, -LF, and -LL in any combination. By default, the listing options apply to both copied and deleted objects. To list only deletions, use -LDL with -L, -LD, -LF, or -LL.

-L	List	all	objects	25	thev	are copied.
-11	LIBU	all	ODJecos	aw	uncy	are copied.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

WARNING

Five conditions always terminate execution:

- You attempt to use the network root or node entry directory as a target, without specifying a merge.
- An error occurs in reading the top level of the source tree.
- You attempt to create an existing directory (if the target is an existing directory, you must specify -R or -M).
- The logical volume containing the target directory is full.
- A quit or stop fault occurs in this process.

EXAMPLES

- 1. \$ cpt /com /com.backup -r Copy the directory tree "/com" to "/com.backup" replacing the existing "/com.backup" tree.

CRD (CREATE_DIRECTORY)

CRD (CREATE_DIRECTORY) -- Create a directory.

FORMAT

CRD pathname ...

CRD creates a directory with the specified pathname.

ARGUMENTS

pathname

(required)

Specify the subdirectory name to be created. Multiple pathnames are permitted. The new directory receives its parent directory's initial ACL.

OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

CREFS (CROSS_REFERENCE_SYMBOLS) -- Cross-reference symbols in a file.

FORMAT

CREFS [-F] [pathname ...]

CREFS produces a cross-referenced list of the symbols in each of the named files, and writes each list to standard output. A symbol is a string of letters, digits, underscores, and dollar signs and must begin with a letter. The list contains every symbol in the file in alphabetical order, followed by the numbers of the lines in which the symbol appears.

Symbols of more than 32 characters are truncated.

ARGUMENTS

pathname

(optional)

Specify input file. Multiple pathnames and wildcarding are permitted: separate names with blanks.

Default if omitted: read text from standard input.

OPTIONS

If the option is not specified, CREFS treats uppercase and lowercase letters as different characters, and places uppercase letters before lowercase letters in the alphabetical sort.

-F Treat all input text as lowercase while cross-referencing.

-K key_fileOnly the words listed in 'key_file' are cross-referenced. These words must be listed one per line.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

To find all occurrences of certain variables in the program "cycle," type:

\$ crefs cycle

CREFS can also be used in conjunction with other commands to produce more refined results. For instance:

\$ crefs cycle | tee cycle.all | fpat wheel spoke axle > cycle.some

The output file "cycle.all" contains a list of all the symbols in the program, with references to the line containing them. The output file "cycle.some" contains only the lines with references to the three variables named: wheel, spoke, and axle.

CRF (CREATE_FILE)

CRF (CREATE_FILE) -- Create a file.

FORMAT

CRF pathname...

CRF creates a zero-length file with the specified pathname. The new file receives its parent directory's initial ACL for files. (See the ACL command description for more information.) The file type is set to OBJECT and the file is made permanent. The type UID is set to nil.

This command is most useful for system-level debugging. Use the Display Manager editing capability to create normal text files.

ARGUMENTS

pathname

(required)

Specify file to be created. Multiple pathnames are permitted, separated by blanks.

OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ crf my_file

CRL (CREATE_LINK) -- Create a link.

FORMAT

CRL linkname object_name ... [-R]

CRL is used to create links. Links normally serve two functions: as a shorthand way of specifying objects with lengthy (and frequently recurring) pathnames and as static pointers to other objects.

Links cause the Shell to redirect a pathname to another object. In effect, links allow you to take a detour from one part of the naming tree to another.

ARGUMENTS

linkname

(required)

Specify the link's name and location.

object_name

(required)

Specify the object to which the link points.

Multiple linkname/pathname pairs are permitted.

OPTIONS

-R

Replace an existing link. Use this option to change a link's object name.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ crl bugs /maintenance/reports

Create a link called "bugs" in the current working directory.

Now, when you use "bugs" in a pathname, the command Shell substitutes the text "/maintenance/reports". Therefore, the pathname:

bugs/sticky_cursor

refers to the same file as the pathname:

/maintenance/reports/sticky_cursor

CRP (CREATE A PROCESS)

CRP (CREATE_A_PROCESS) -- Create a process on a remote node.

FORMAT

CRP [command line] -ON node_id [options]

CRP creates a process on a remote node.

ARGUMENTS

command line

(optional)

Specify command line to be executed by the remote process. If the command string contains embedded blanks, enclose it in quotation marks.

Default if omitted: execute /COM/SH (the Shell).

OPTIONS

The following option which specifies the remote node is REQUIRED:

-ON node_id Specify either hexadecimal node_id or entry directory name of any volume on the selected remote node.

One of the following options may be specified (default option is indicated by a "(D)"):

- -CP (D) Create a remote process running with standard streams connected to the current window. Not valid if -CPO or -CPS is specified. You may stop these processes by typing an EEF (usually CTRL/Z) in the process input pad.
- -NWP Do not create a window pane legend indicating that the local window is connected to a remote process. Use with -CP option only.
- -CPO

 Create a remote process without a connection to the current window, and an identity of 'user.none.none'. Not valid if -CP or -CPS is specified. To stop these processes, you must first create a visible remote process running the Shell, then issue the SIGP command to stop the background process.
- -CPS

 Create a remote process without a connection to the current window, and an identity of 'user.server.none'. Not valid if -CP or -CPO is specified. To stop these processes, you must first create a visible remote process running the Shell, then issue the SIGP command to stop the background process.
- -N name Specify the name of the remote process. If this option is not specified, the default is "user id.node_id". This allows remote processes to be traced to their originator.

-LOGIN name [password]

Specify the login sequence for the remote process on the command line. If the password is omitted, the system prompts you for it. A null (zero-length) password is specified by the null string ".

Normally, -LOGIN appears with -CP. You may, however, use -LOGIN with -CPO and -CPS as well. If -LOGIN is specified with either -CPO or -CPS, then the identity of the created process is the same as that of the caller's (as opposed to 'user.none.none' or 'user.server.none', respectively). This means that -CPO and -CPS are identical if -LOGIN is also specified.

If you use -LOGIN with -CPO or -CPS, you must place both the name and the password on the command line. No prompting is available in this case.

-ME

-ME is typically specified instead of -LOGIN. If -ME is specified, then the created process on the remote node inherits the caller's working directory, naming directory, home directory text string and SID. In some sense, this is similar to popping up another Shell except that the process is running on another node. If -ME is specified with either -CPO or -CPS, then the identity of the created process is also that of the caller's (as opposed to 'user.none.none' or 'user.server.none', respectively). This means that -CPO and -CPS are identical if -ME is also specified.

EXAMPLES

\$ crp -on 532 -login joe

Create a process on node 532 running the Shell, and log in with the user id "joe"

\$ crp -on Oaef -me

Create a process on node AEF running the Shell, and inherit the current process state information.

CRPAD (CREATE_PAD) -- Create a transcript pad and window.

FORMAT

CRPAD [pathname] [options]

CRPAD creates a transcript pad, copies a file (or standard input) into that pad, and then opens a window into the pad. This new pad is NOT related to the transcript pad attached to processes running the Shell; it is for viewing file contents only. This is primarily useful for displaying output being produced inside a pipeline without interrupting the flow of control in the pipe.

Transcript pads are not editable. If you wish to place a file in a pad for editing, use the <EDIT> key or the DM command CE (CREATE_EDIT).

CRPAD -IN behaves differently. This creates an edit pad and lets you create whatever text you want. When you close the edit pad (with WC or CTRL/Y), that text is copied to standard output.

ARGUMENTS

pathname

(optional)

Specify the file to be copied into the pad. Not valid if -IN is

present.

Default if omitted: copy standard input.

OPTIONS

-IN[PUT]

Copies data from a temporary edit window to standard output.

Not valid if -TEE or -PN are specified.

-PN pathname

Specify a pathname for the pad. If you specify a pathname, the pad is saved in that file. Note that you can also save the pad after it is created by using the DM command PN (PAD NAME).

-TEE

Copy output to standard output in addition to the new pad.

EXAMPLES

- 2. \$ fpat -p '256-' <phone.book | crpad -tee | srf >phone.book.local
 Display the intermediate results
 in a pipeline.
- 3. \$ crpad -input | srf | crpad

Create an edit pad. When the pad is closed, sort the text edited and display it in a transcript pad.

CRRGY (CREATE_REGISTRY) -- Create or modify network registry.

FORMAT

CRRGY [options]

CRRGY allows you to create a network registry, create a local registry on the master node, or change registry sites. While all the functions of this command are described below, it is unlikely that you will be able to manipulate the network registry unless you are the network administrator for your network (due to ACL settings on the registry files). You may, however, have access to the local registry for your particular node to manipulate as you like. For complete information about both network and local registries, see Administering Your DOMAIN System.

OPTIONS

At least one of the following options must be specified.

-R rgypath

Specify the pathname of the new registry object you want to create or the registry on which you want to operate. If you provide just a node name (//node), CRRGY creates the registry "//node/registry/rgy_master", or CRRGY uses the file //'this_node'/registry/registry to locate the master registry. If you omit a pathname, CRRGY creates "//'this_node'/registry/rgy_master." We recommend that you specify only a node name.

Default if omitted: create //this_node/REGISTRY/RGY_MASTER

-S site...

Create new site(s). 'site' specifies the pathname of the registry directory you want to create. Multiple sites are permitted, separated by blanks. Files created at the sites are filled with default identities. The new sites are included in the new registry master file. If you do not specify -S, and you are creating a registry, "//'this_node'/registry/rgy_site" is used. Sites may be located anywhere in the network. This option is not valid for local registries.

-A site...

Add site name(s) to the existing registry specified by the -R option. This command creates a directory at the site, copies all the registration data files to the directory, and adds the site names to the master file. This option is not valid for local registries.

-D site...

Delete site name[s] in the existing registry master file specified by the -R option. The master registry file reflects the change but the registry file copies ('/registry/registry') do not. This option is not valid for local registries. The site directories are NOT deleted; their names are removed from the registry file.

-LOC

Create a local registry. The only valid pathnames for a local registry are the following default names:

CRRGY (CREATE_REGISTRY)

"//'node'/registry/local_registry" for the master, and

"//'node'/registry/local_site" for the site.

Thus, you should only specify a root name ("//node") with the -R option and let all other defaults apply.

-EX

Specify that the given sites already exist. Do not create the sites specified, just include their names in the registry master file. You may use this option to create a replacement registry master file that points to existing sites or to add existing site names to an existing master file (-A). No check is made to ensure that the sites really do exist. The site names must be full pathnames when you use this option.

-SEC

Set secure passwords to a minimum length of six characters. This option is not valid for local registries.

-COUNT n

Specify the number of slots (n) to create in a local registry account file. The default number is 25; minimum is 5. This option requires -LOC. Nodes serving diskless partners should have enough entries to allow slots for the users of the diskless nodes.

-DAYS n

Specify number of days (n) during which an entry in a local registry account file will remain valid after login. The default number is 21; maximum is 60. This option requires -LOC.

EXAMPLES

\$ crrgy -loc -count 50 -days 30

Create a new local registry with spaces for 50 account entries which will each be valid for 30 days after login. Previous local registry accounts are lost.

For detailed examples of network registry creation, see Administering Your DOMAIN System.

CRSUBS (CREATE_SUBSYSTEM) -- Create a protected subsystem.

FORMAT

CRSUBS subsystem_name

CRSUBS is used to create a protected subsystem.

A protected subsystem is a set of programs and data files. The programs are called the "managers" of the protected subsystem; the data objects which they manage are said to be "owned" by the subsystem. Protected subsystems allow you to define exactly how a file can be accessed.

CRSUBS creates a protected subsystem by creating the "subsystem shell" and putting it in the directory /SYS/SUBSYS. Once there, the subsystem shell can be invoked using the ENSUBS command. The access control list for the directory /SYS/SUBSYS determines who can create new subsystems: whoever has 'a' (append) rights to the directory will be able to create new subsystems.

The access control list on the file /SYS/SUBSYS/subsystem_name determines who can enter the subsystem 'subsystem_name'. Whoever has read and execute rights to it can enter the subsystem. This capability should be restricted generally to the creators of the subsystem or to the system administrators.

For more information on protected subsystems, see the DOMAIN System User's Guide.

ARGUMENTS

subsystem_name (required)

Specify name of subsystem to be created. This file will contain the "subsystem shell" and will reside in the /SYS/SUBSYS directory.

EXAMPLES

\$ crsubs newsubs

Create the subsystem "newsubs".

CRUCR (CREATE_UCR) -- Create a User Change Request form.

FORMAT

CRUCR

We appreciate feedback from users of our systems. To make the feedback process as easy as possible, we provide User Change Request (UCR) forms for you to record comments and suggestions.

CRUCR enables you to complete user change request forms on line. CRUCR assigns each form a unique number and stores the completed form in /SYS/UCR. /SYS/UCR may be a link.

The first time you execute CRUCR on a node, the command prompts you for information about your site and creates the object file /SYS/UCR/DMMPROCxxxx, where xxxx is your node ID. This file contains the default UCR form that is used from then on. You may edit this template to add information to subsequent UCR forms.

Submit UCRs to us on floppy disks or as line-printer listings, using a separate UCR for each distinct problem that you report. If you are sending in a program, please use a floppy disk. Mail the material to:

Apollo Computer Inc. 330 Billerica Road Chelmsford, MA. 01824

Attn: UCR Administrator, Technical Operations

CRUCR does not require any arguments or options.

CSR (COMMAND_SEARCH_RULES) -- Set or display command search rules.

FORMAT

CSR [directory ...] [-A dir_name]

Command search rules determine which directories the Shell examines to find commands. CSR lets you display or change this list. If a new Shell is invoked *inside the current process*, or a new Shell script run, the subordinate Shell inherits the search rules of the parent Shell. Note that this does *not* apply to "shells" created by the <SHELL> key, since that key actually creates a new, separate process. Its Shell receives the default search rules described below.

By default, the Shell looks for commands in this order:

- 1. Your working directory ("."), or the directory specified by the command's pathname.
- 2. Your personal command directory, ~COM (the COM subdirectory of your naming directory).
- 3. The system command directory, /COM.

Refer to The DOMAIN System User's Guide for a detailed discussion of command search rules.

Specifying CSR without arguments or options displays the current command search rules.

ARGUMENTS

directory

(optional)

Specify new command search sequence. Multiple directory pathnames are permitted; separate names with blanks. The Shell will search the directories in the order that you specify.

Default if omitted: display current search rules unless -A is specified.

OPTIONS

-A dir name

Append the specified directory name(s) to the existing command search sequence. This allows you to add a new directory to the end of the list without retyping the entire list. Multiple directory pathnames are permitted; separate names with blanks.

EXAMPLES

1. \$ csr . ~com /com Display current search rules.

CSR (COMMAND_SEARCH_RULES)

- 2. \$ csr ~com //us/myproj/com /com
- 3. \$ csr -a ~com/special_commands
- Set new search sequence by adding an additional command directory.

Append the directory ~com/special_commands to the current list of directory names. CTNODE (CATALOG_NODE) -- Catalog a node in the network.

FORMAT

CTNODE [node_name node_id ...] [options]

CTNODE informs the local node that a remote node exists, thereby enabling network file access to the remote node. The command catalogs the node_name in the local copy of the network root directory as the entry directory for the remote node. In other words, CTNODE adds the directory //node_name to your copy of the network root directory. For information on deleting a node_name entry, see UCTNODE (UNCATALOG_NODE).

We assign a node ID to every node during the manufacturing process. To find out the node ID of a node, type the following command at its keyboard:

\$ lcnode -me

At SR9.0, CTNODE supports the ability to merge information from another node's network root into your own, or any other node's network root. The merge options (-MD and -MS) add the entry for a node to the target provided the entry does not already exist and the source has exactly one entry for that node. In the case of 1 source and 1 target entry for a node which match, those entries are assumed to be correct. All other cases are considered to be ambiguous and the "confusion resolution protocol" is invoked.

This "confusion resolution protocol" first attempts to verify the correct entry name with the node itself. If the node is available, then the reply from the node is cataloged regardless of whether '-MD' or '-MS' is used. This is because an answer from the node itself is assumed to be the truth.

If the node is unavailable to resolve an ambiguity, then the entry which contains the most recent UID (latest time stamp portion of the UID), is used. In this case, existing entries in the target directory are only updated if the '-MS' option is used.

ARGUMENTS

node_name

(optional)

Specify the entry directory name of the node you wish to catalog. If the 'node_id' argument is specified, then 'node name' is required.

Default if omitted: must use -N, -UPDATE, or -FROM

node_id
(optional)

Specify the hexadecimal ID of the node you wish to catalog. The node must be connected to the network when this command is executed. If the 'node_name' argument is specified, then 'node_id' is required.

Default if omitted: must use -N, -UPDATE, or -FROM

Multiple name/ID pairs are permitted.

If neither -N, -UPDATE, or -FROM is specified, then the 'node_name' and 'node_id' arguments are required. The -N, -UPDATE, and merge options work only for remote nodes running AEGIS operating system software release 5.0 or later.

-ROOT

Catalog 'node_name' as the entry directory name for 'node_id' in both the master network root directory and the local copy of the network root directory. This option is valid only if the 'node_name' and 'node_id' arguments are specified. This option is not valid if the -N option is specified.

-N node id...

Copy the entry directory name from the network root directory of the specified remote node, to the network root directory of the local node. You do not need to know the entry directory name. However, you must specify the node ID of the remote node. Multiple node_id's may be specified. Use this option instead of the 'node_name'/'node_id' argument pair. This option is not valid if the -R option is specified.

-UPDATE

Obtain a list of nodes currently responding to a network inquiry and perform the same operation as "-N" for each node. Names are replaced with the most current version, if they already exist in your local copy of the network root directory, and new names are added.

-FROM //node ...

Look in the specified list of network root directories for the names to add to the target network root, or use this network root as the source for names to merge into the target network root. Wildcards may be used to specify source node names.

-MD

Used with '-FROM'. Merges all names in the source network root into the target network root. Preference is given to existing names in the target if there are unresolved conflicts (see the discussion of "confusion resolution" above).

-MS

Same as -MD, except that preference is given to entries in the source network root when there are unresolved conflicts (see the discussion of "confusion resolution" above).

-ON //node ...

Catalog names in the network root of the specified nodes instead of the local network root. Wildcards may be used to specify target node names.

-R

Replace cataloged names if they already exist. An error occurs if you do not specify this option and try to add a node_name that has already been cataloged (unless you are using "-UPDATE").

-L

List node names as they are cataloged.

-IDUPL

Ignore entry (suppress error messages) if name already exists in the target.

-LC

List invocations and resolutions of the "confusion resolution protocol".

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1. \$ ctnode os 21 Add the node whose ID is 21 and whose entry directory name is "os" to your node's catalog.

2. \$ ctnode -update Bring your node's catalog up to date with any new nodes on the network.

3. \$ ctnode os eve -from //master Copy names "os" and "eve" from the network root on //master.

4. \$ ctnode os 21 -on //a?* Add node ID 21 with the name "os" to the network root of all nodes whose names begin with "A".

5. \$ ctnode -md -from //os Merge network root of OS into local network root, resolving conflicts. CTOB (CATALOG_OBJECT) -- Catalog an object.

FORMAT

CTOB pathname uid_hi uid_low

CTOB assigns a pathname to an object that has a known unique identifier (UID). CTOB catalogs the pathname and associated UID in the naming tree. This command is primarily for system-level debugging.

ARGUMENTS

pathname

(required)

Specify assigned pathname.

uid_hi

(required)

Specify the high portion of the UID as a 32-bit hexadecimal

number.

uid_low

(required)

Specify the low portion of the UID as a 32-bit hexadecimal

number.

OPTIONS

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ ctob lastfile 10A0BAAD 60000102

CVT_REC_UASC (CONVERT_RECORD_UASC) -- Convert file types.

FORMAT

CVT_REC_UASC source_pathname [target_pathname] -OT type [options]

CVT_REC_UASC converts files from type "rec", "hdru", or "uasc" to files of type "rec", "hdru", or "uasc". It functions on nodes running software release 4.1 and later.

ARGUMENTS

source_pathname

(required)

Specify the file to be converted.

target_pathname

(optional)

Specify file to be created. An error occurs if this file already exists (see -R below). The target_pathname may be derived. If target is a directory, the source file is converted and placed in that directory.

Default if omitted: the converted file becomes

'source_pathname' and the original file is renamed 'source_pathname.CBAK'.

-OT type

(required)

Specify type of file to be created ('target_pathname'). Choose one of the following for 'type': "rec", "hdru", or "uasc."

Wildcards in pathnames associated with this command are permitted.

OPTIONS

-R

Replace 'target_pathname' if it already exists.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ 1d -a
 List current files in specified directory and their types.

Directory "/larry/cvt_rec_uasc_examples":

sys type	type uid		current	attr	rights	name
file	rec	1	42	P	pndwrx	a
file	rec	1	42	P	pndwrx	ъ
file	rec	1	44	P	pndwrx	С

3 entries, 3 blocks used.

2. \$ cvt_rec_uasc ?* -ot uasc -nq \$ ld -a Convert all files to type uasc; suppress wildcard queries.

Directory "/larry/cvt_rec_uasc_examples":

sys type	type uid	blocks used	current length	attr	rights	name
file	uasc	1	37	P	pndwrx	a
file	rec	1	42	P	pndwrx	a.cbak
file	uasc	1	38	P	pndwrx	b
file	rec	1	42	P	pndwrx	b.cbak
file	uasc	1	40	P	pndwrx	С
file	rec	1	44	P	pndwrx	c.cbak

6 entries, 6 blocks used.

3. \$ cvt_rec_uasc [a-c] =.x -ot rec -nq \$ ld -a Convert files named "a" "b" and "c" to type rec and write them to "a.x" "b.x" and "c.x"

Directory "/larry/cvt_rec_uasc_examples":

sys type	type uid	blocks	current length	attr	rights	name
file	uasc	1	37	P	pndwrx	a.
file	rec	1	42	P	pndwrx	a.cbak
file	rec	1	42	P	pndwrx	a.x
file	uasc	1	38	P	pndwrx	ъ
file	rec	1	42	P	pndwrx	b.cbak
file	rec	1	42	P	pndwrx	b.x
file	uasc	1	40	P	pndwrx	С
file	rec	1	44	P	pndwrx	c.cbak
file	rec	1	44	P	pndwrx	C.X

⁹ entries, 9 blocks used.

\$

DATE -- Display the current date and time.

FORMAT

DATE [options]

DATE prints the current system date and time. It requires no arguments or options. If no options are specified, the date is displayed as shown in Example 1 below.

The hardware date and time may be set with the Shell command CALENDAR.

OPTIONS

-Y Display year as YYYY.

-MD Display month and day as MM/DD.

-T Display time in 24-hour format (HH:MM:SS).

-D Display year, month, and day.

EXAMPLES

\$ date
 Tuesday, June 9, 1981 4:20:15 pm (EDT)

2. \$ date -t 15:36:14

\$ date -d
 1983/08/08

DCALC (DESK_CALCULATOR) -- Evaluate logical and arithmetic expressions.

FORMAT

DCALC [-H] [pathname...]

DCALC mimics the features of a desk calculator, evaluating both logical and arithmetic expressions.

ARGUMENTS

pathname

(optional)

Specify input file containing expressions to be evaluated, one expression per line.

Default if omitted: read standard input; stop with CTRL/Z

OPTIONS

If no options are specified, all operations are decimal-based.

-H

Specify hexadecimal operations.

Expressions

Input expressions can be simple arithmetic expressions or variable assignment expressions. DCALC writes the value of each evaluated expression on standard output. Variables hold temporary values, which DCALC does not automatically write.

Expressions may include any of the operators listed below (in order of precedence):

- 1. + unary plus and negation operators. These may only appear at the start of an expression or within parentheses.
- 2. << >> logical left and right shift
- 3. ** exponentiation
- 4. * / % multiply, divide, modulo (remainder)
- 5. + add, subtract
- 6. == equal to
 - != not equal to
 - > greater than
 - >= greater than or equal to

< less than

<= less than or equal to</pre>

7. ! unary logical not

8. | logical or

& logical and

logical xor

Relational operators return the value 1 for true and 0 for false. DCALC performs operations in double precision floating point, except for logical operators listed as items 2 and 8 above, which use 32-bit integers.

Variables

Expressions may include previously declared variables. Use this format to declare a variable:

name = expression

- A variable name must begin with a letter and may consist of any combination of letters and digits.
- DCALC does not automatically print replacement expressions, because they usually contain temporary values.

Radix Control

You can change the default base for input or output using ibase (input base) and obase (output base) statements. For example,

ibase = 2

obase = 16

causes DCALC to interpret input in binary and print results in hexadecimal.

To set an individual number's radix, precede it with the desired radix and a pound sign. For example,

16#100

specifies the hexadecimal number 100 (equals 256 in decimal).

DCALC (DESK_CALCULATOR)

EXAMPLES

Your input: DCALC output:

1. 10 + (-64 / 2**4) 6

2. temp = 2#101 temp == 5 1 (true)

3. ibase = 16 obase = 2 11 + 28 111001 1a + 0f 101001

(Note that when you type a hexadecimal number that begins with a letter, you must precede it with a zero.)

4. ibase = 16 numa = 100 numb = 100 numa + numb 512 DEBUG -- Invoke the Language Level Debugger.

FORMAT

DEBUG [options] {-PROC process_name | target [args...]}

The Language Level Debugger provides an easy way to debug high-level language programs (Pascal, FORTRAN, C). When you invoke DEBUG, it splits the current window into two sections, or "windowpanes". The DEBUG session runs in the top pane, and your program runs in the bottom pane. These windowpanes disappear when you terminate the DEBUG session.

NOTE: DEBUG gives you two different ways to invoke the target. If you choose to invoke DEBUG and specify a target name, all DEBUG options MUST appear before the target name. DEBUG looks for its own options until it finds the target pathname, and then stops looking. (This permits a target program to have an option with the same name as a DEBUG option, if necessary.)

If, on the other hand, you invoke DEBUG using the -PROC switch (described below), then all DEBUG options must appear before the -PROC switch, and NO target name OR target options are allowed. The target's invocation, with the target's own arguments, are given separately in the other process.

ARGUMENTS

One of the following two arguments is required on the command line.

target [args...]
(optional)

Specify the pathname of the file containing the program you wish to debug, plus any arguments that the program may require.

-PROC process_name

(optional)

Perform explicit cross-process debugging. This allows you prime DEBUG to watch for target invocation in an already-existing process (specified by 'process_name'). After you invoke DEBUG with this option, it will watch the given process for the invocation of a program. When the invocation of the target program occurs, DEBUG will wake up and take control of the target program, and give you the regular DEBUG access to it.

This feature is especially useful when the normal invocation of DEBUG and a target perturbs the environment enough to make a bug disappear. In this case, creating a new process for DEBUG and directing it to watch the old process will guarantee that the invocation of the target in the old process will have exactly the same state as it had before you tried to debug it.

OPTIONS

Note that all DEBUG options MUST PRECEDE the pathname of the target program.

-WPn

Specify size of DEBUG windowpane from 10% to 90%. 'n' may equal 10, 20, 30, 40, 50, 60, 70, 80, or 90. Default if omitted: DEBUG creates a windowpane in the top 50% of the window.

-NWP

Do not create DEBUG windowpanes. Instead, DEBUG will perform input and output operations using the error input and error output streams in your transcript pad.

-NC

This option prevents DEBUG from copying the object file. Instead, the actual file is mapped for writing and executed directly.

-SRC

-SRC_T

-SRC_R

These options cause DEBUG to display the source file(s) which were used to make the target program being debugged. The -SRC option lets DEBUG choose where the source will be displayed. The -SRC_T option makes DEBUG put the source at the top of the window; the -SRC_R option makes DEBUG put the source on the right-hand side of the window.

-SDIR pathname

This option provides alternative directory pathnames for finding the source file(s), when one of the -SRC options is used. It may be given any number of times on the command line. (When the source was compiled, the full pathname was remembered. If DEBUG cannot find the source file using that pathname, the source filename is extracted from the full pathname and is appended to each SDIR pathname. Then, DEBUG tries each of these pathnames as well.)

-R[EAD] pathname

This option causes DEBUG to read the file specified by 'pathname' and process each line as a DEBUG command as DEBUG is invoked. It may be given only once on the command line. (As a regular DEBUG command, it may be given more than once.)

-SMAP

This option generates a brief section map of the loading of the target program.

-SET arg_string

Set debug variable prior to invoking the target program. 'arg_string' is the body of a valid Set command. The string must be quoted if it contains spaces, so that the command line parser sees it as one argument. See Example 2 below. If you submit a -SET option which does not have an assignment operator (=, :=) in 'arg_sring', the Set command will be interactive as it is during normal DEBUG operation.

DEBUG COMMANDS

Once DEBUG is invoked, it recognizes the following commands. See the DOMAIN Language Level Debugger Reference for complete details about each command.

A[RGS] - Display the arguments of an active routine.

B[REAKPOINT] - Set a breakpoint.

DEF[INE] - Create or change a substitution string.

D[ELETE] - Delete a breakpoint, macro, or substitution string.

DES[CRIBE] - Print information about a variable's definition.

ENV[IRONMENT] - Display/change the assumed symbolic routine name.

E[XAMINE] - Examine a variable.

EXI[T] - Leave the debugger.

G[0] - Run the target program.

H[ELP] - Display brief information about the commands.

IF - Evaluate a conditional expression and do commands.

J[UMP] - Go to a labeled statement in an action list (command list).

L[IST] - List routines, breakpoints, macros, substitution strings

and debugger variables.

M[ACRO] - Create or change a macro.

P[RINT] - Print the value of one or more expressions.

Q[UIT] - Leave the debugger.

R[EAD] - Read a DEBUG command file.

SDIR - Provide alternate pathnames for source file display.

S[ET] - Set a variable to a new value.
SH[ELL] - Invoke a Shell under the debugger.

STCODE - Display information about a given status code.

ST[EP] - Step the target program one statement.

TB - Traceback the current user calls.

VA - Show the Virtual Address of current program location,

one named routine, or a list of variables.

NOTE:

Certain options chosen in the compilation phase of program development (i.e., -NDB/-DB/-DBS/-DBA) may influence your ability to access those programs via DEBUG. See the appropriate compiler manual for details.

DEBUG MACROS AND VARIABLES

DEBUG allows you to create simple variables, macros, or definitions which are local to DEBUG. Futhermore, there are a handful of debugger names which DEBUG recognizes, and which can help you control how DEBUG works. If you want to use one or more of these, it is convenient to establish them in a DEBUG startup file in the "user_data" subdirectory of your login home directory. DEBUG will look for a file named "startup_debug" and execute any commands that are present.

The special DEBUG names are as follows. DEBUG names are case insensitive.

Macros:

'cr

This macro, if present, will be invoked whenever you press the RETURN key by itself to a DEBUG prompt. One of the most useful values for the "carriage return" macro is "macro 'cr [st]" to step through your program a statement at a time. This macro is optional. If you define it, and then want to undo it, define it as "macro 'cr []".

Variables:

Note that each of these variables is optional. Furthermore, if you set one, and then later want to "unset" one, set its value to be less than zero.

 $\verb"max_var_len"$

This variable sets a limit on the number of characters to be displayed for any variable on an Examine, Set, DEScribe, or VA command. For example: "set 'max_var_len = 60" will make DEBUG stop displaying routine\variable-name at the 60th character. Dropped characters are flagged by "...".

'max_array_dim

This variable sets a limit on the number of elements to be displayed for any array or sub-array. For example, "set 'max_array_dim = 4" will cause DEBUG to display only four elements for any array access.

'max_string_len

This variable sets a limit on the number of characters displayed when a character array is printed with the Print command. This is different from 'max_array_dim, and applies only to the Print command. If you Examine the same character array, 'max_array_dim applies.

'src adjust

During source display, when you reach a line which requires a new page of source, the current line becomes the top line of the window, giving no prior context. The 'src_adjust variable allows you to specify that some number of prior lines be shown. For example, if you "set 'src_adjust = 4", then 4 lines prior to the current line will be shown at the top of the source display window.

'src try bak

This variable, if present and greater than zero, will cause DEBUG to try to find a ".bak" version of a source file if it finds that the current version of a source file does not match the version information it has in the object file.

EXAMPLES

1. \$ debug -src_r my_prog

Debug MY_PROG, displaying the source file in the right side of the process window.

2. \$ debug -src -set "'max_array_dim = 8" my_prog

Set the 'MAX_ARRAY_DIM variable as you invoke DEBUG; allow DEBUG to select the most appropriate type of source display.

DLDUPL (DELETE_DUPLICATE_LINES) -- Strip repeated lines from a file.

FORMAT

DLDUPL [-C] [pathname ...]

DLDUPL reads the input file(s), comparing adjacent lines. Second and succeeding copies of repeated lines are removed; the remaining lines are written to standard output.

ARGUMENTS

pathname

(optional)

Specify input file. Multiple file names permitted; separate

names with blanks.

Default if omitted: read standard input

OPTIONS

-C

Write number of occurrences of each line to standard output.

EXAMPLES

Suppose you have two dictionary files. To create one dictionary file containing the words from both, use:

\$ srf -m dict1 dict2 | dldupl >dict.new

This merges the words from the two files (SRF -M), then deletes any duplicate words and saves the result in the new dictionary.

DLF (DELETE_FILE) -- Delete one or more files.

FORMAT

DLF [pathname...] [options]

DLF deletes the file(s) specified. To delete objects other than files, see DLL (DELETE_LINK) and DLT (DELETE_TREE).

ARGUMENTS

pathname

(optional)

Specify file to be deleted. Multiple names and wildcarding are

permitted; separate names with blanks.

Default if omitted: read names from standard input

OPTIONS

-F Force file deletion if you have owner rights, even if you don't

have delete rights.

-L List names of deleted files.

-DU Delete when unlocked. If the object to be deleted is locked when

DLF is invoked, the delete operation will be performed when the

object is unlocked.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ dlf mary.bak -L
(file) "mary.bak" deleted.

DLL (DELETE_LINK) -- Delete a link.

FORMAT

DLL pathname ... [options]

DLL deletes a link. After execution of this command, the link is no longer available for use.

ARGUMENTS

pathname

(required)

Specify pathname of the link to be deleted. Multiple pathnames and wildcarding are permitted; separate names with blanks.

OPTIONS

-L

List name(s) of link(s) as deleted.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ dll bugs

\$

Delete the link "bugs" from the current working directory.

DLT (DELETE_TREE) -- Delete a tree.

FORMAT

DLT pathname ... [options]

DLT deletes the directory named by the pathname, and all its descendants in the naming tree.

ARGUMENTS

F	athn	ame

(required)

Specify directory or link to be deleted. If "pathname" is a directory, DLT deletes the directory and all subordinate objects (subdirectories, files, and links). If a link, DLT deletes the link name, but has no effect on the files and directories named by the link. Multiple pathnames and wildcarding are permitted.

OPTIONS

-L List files, links, and directories as they are deleted.

-LD List directories as they are deleted.

-LF List files as they are deleted.

-LL List links as they are deleted.

Force object deletion if you have owner rights, even if you don't

have delete rights.

-DU Delete when unlocked. If the object to be deleted is locked when

DLT is invoked, the delete operation will be performed when the

object is unlocked.

-PR pathname

Preserve specified pathnames.

-LD, -LF, amd -LL may be combined to create the type of listing you desire.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ dlt april_backup may_backup

Delete the two directory trees specified.

DLVAR (DELETE_VARIABLE) -- Deletes all of the specified variables.

FORMAT

DLVAR var_name ...

The DLVAR command deletes the variable(s) specified. If a variable had another value at a higher level of invocation, the variable is restored to that value.

ARGUMENTS

var_name ...

(required)

Specify the variable name to be deleted. Multiple names are permitted, separated by blanks.

DMTVOL (DISMOUNT_VOLUME) -- Dismount a logical volume.

FORMAT

DMTVOL device[unit] [log_vol_number] [pathname] [options]

DMTVOL dismounts a logical volume that was previously mounted with the MTVOL (MOUNT_VOLUME) command. After the volume has been dismounted, it is unavailable for further access.

ARGUMENTS

device

(required) Specify the type of disk on which the volume resides: "W" for

a Winchester disk, "S" for a storage module, or "F" for a floppy

disk.

unit

(optional) Specify a unit number (0 or 1 only) for the device, if necessary.

For example, "S1" denotes storage module unit 1.

Default if omitted: 0 (zero)

log_vol_number

(optional) Specify the number of the logical volume to be dismounted.

Default if omitted: 1

pathname

(optional) Specify the entry directory of the logical volume. If you include

this argument, DMTVOL dismounts the volume and uncatalogs its entry directory. If you omit it, DMTVOL dismounts the

logical volume, but retains its name in the naming tree.

Default if omitted: see above

OPTIONS

-FU Forcibly unlock any locked objects, then dismount the volume.

If you omit this option, the dismount fails if the volume contains

any locked objects.

-NW No_write -- Prevents DMTVOL from trying to write to the

disk during the dismount. Normally, writing to the disk saves current information. However, if the disk was removed prior to

dismount, this option should be used.

EXAMPLES

1. \$ dmtvol s 2 Dismount storage module unit zero, \$ logical volume 2, and leave its name in the naming tree.

2. \$ dmtvol f /floppy Dismount floppy unit zero, logical volume 1, and delete its name from the naming tree.

DSPST (DISPLAY_PROCESS_STATUS) -- Display process status graphically.

FORMAT

DSPST [options]

DSPST displays process statistics in a graphical, bar-chart fashion within the current process window. The chart is updated periodically (see -R below). The default action of this command is to display the brief OS process list, all user processes and all I/O information in a font size automatically selected based on window size.

While DSPST is running, the following keys are interpreted as follows;

All keyboards:

Move to top
Move to bottom
Exit
Exit
Exit and save current image

880 keyboard only:

Boxed	up arrow	Scroll	forward 1/2 window
Boxed	down arrow	Scroll	backward 1/2 window
F2		Scroll	backward 1 line
F3		Scroll	forward 1 line

Low-profile keyboard only:

Boxed up arrow	Scroll backward 1/2 window
Boxed down arrow	Scroll forward 1/2 window
Shifted up arrow	Scroll backward 1 line
Shifted down arrow	Scroll forward 1 line
EXIT or ABORT	Exit
SAVE	Exit and save current image

OPTIONS

Default options are indicated by "(D)."

-R n		Specify that the display should be repeatedly updated every 'n' seconds. If this option is omitted, the display is updated every 4 seconds.
-P		Show process info.
-L1		Show OS and user process info.
-OS	(D)	Show brief OS and full user process info.
-M		Show missing cpu time in addition.
-IO	(D)	Show I/O statistics.

-A

Show all information (same as -L1 -IO -M).

-N node-spec

Specify remote node whose process stats are to be listed. Node-spec may be the target node's hexadecimal node ID, or the entry name of any volume on that node.

-LARGE

Force use of large font for display.

-SMALL

(D) Force use of small font for display.

EXAMPLES

\$ dspst

Display OS, user process, and I/O status.

\$ dspst -n //fred -large

Display OS, user process, and I/O status for the node named //fred using the large font.

ED (EDIT) -- Invoke line editor.

FORMAT

ED [-N] [pathname]

ED invokes the line editor. Input text and editing commands are read from standard input. While you may use ED to create text files interactively, it is better suited for use in programs and scripts. Use the <EDIT> key or the DM command, CE, to create and edit files interactively.

NOTE: There is a homonymous DM command: ED -- Delete character preceding cursor. See the ED command description in the DM chapter for details.

ARGUMENTS

pathname

(optional)

Specify file to be edited. ED reads the file into a buffer for editing and remembers its name for future use. ED operates on the buffer copy; changes made there have no effect on the original file until you issue a W (write) command from within ED. Files are limited to 6400 lines.

If the 'pathname' argument is omitted, the edit buffer is empty and no file name is remembered for future use. You will have to specify an explicit file name when you exit the editor.

Default if omitted: see above

OPTIONS

-N

Suppress the printing of line counts by the E (edit), R (read), and W (write) commands.

SUMMARY OF ED COMMANDS

Commands to ED have a consistent format: zero, one, or two line addresses followed by a single-character command, with optional parameters following the command. The general format is:

[line,][line]command parameters

The [line] specifies a line number or address in the current edit buffer. There is usually a useful default for each command (normally the current line) so that you don't need to specify an address explicitly.

Addresses:

17 a decimal number
. the current line
\$ the last line of the file
/pat/ search forward for line containing pat
\pat\ search backward for line containing pat
line+n n lines forward from line
line-n n lines backward from line

Defaults:

- (.) use current line(.+1) use the next line
- (.,.) use current line for both line numbers
- (1,\$) use all lines

Commands:

(.)	Α	Append text after line (text follows)
(.,.n)	Bn	Browse over the next n lines (default n is 22).
		If n is negative, print last n lines before
		current line. If 'B.' is specified, print n
		lines with current line in center of screen.
(.,.)	C	Change text (text follows)
(.,.)	D	Delete text
	E file	Discard current text, enter file, remember filename
	F	Print filename
	F file	Remember filename
(.)	I	Insert text before line (text follows)
(.,.)	Kline	Copy text to new line after specified line
(.,.)	Mline	Move text to line after specified line
(.,.)	Para and many organical	Print text (can be appended to other commands)
	Q	Quit draws habber is residual loss frances in all your
(.)	R [file]	Read file, appending after current line
(.,.)	S/pat/new/GP	Substitute new for leftmost pat (G implies all
		occurrences)
(1,\$)	W [file]	Write file, leave current text unaltered (if
		no file is specified, write to current filename)
(.)	=[P]	Print line number, current line
(.+1)	<cr></cr>	Print next line
(1,\$)	G/pat/command	Execute command on lines containing pat
		(except A, C, I, Q commands)
(1,\$)	X/pat/command	Execute command on lines not containing pat
		(except A, C, I, Q commands)
	#	Comment
	\$n	Read or write temporary buffer, "n".

The error message "?" is printed whenever a command fails or is not understood.

LIMITATIONS

- Files being edited can contain up to 6400 lines.
- When a global search and substitute combination fails, the entire global search stops.
- Problems sometimes occur when removing or inserting NEWLINE characters (via @n), especially in global commands.

ED COMMANDS IN DETAIL

ED accepts commands interactively (from the keyboard) and in a batch-like manner (from script files). To use a script file, substitute the script file name for standard input:

ED [options] [pathname] <script

Command Format

Commands to ED have a consistent format: zero, one, or two line addresses followed by a single-character command, with optional parameters following the command. The general format is:

[line,][line]command parameters

The [line] specifies a line number or address in the current edit buffer. There is usually a useful default for each command (normally the current line) so that you don't need to specify an address explicitly.

Line addresses are formed from the following components:

17 an integer number			
	the current line		
\$	the last line in the buffer		
.+n	n lines past the current line		
n	n lines before the current line		
/pattern/	a forward context search		
\pattern\	a backward context search		

Line numbers can be separated by commas or semicolons; a semicolon sets the current line to the previous address before the next address is interpreted. This feature can be used to determine the starting line for forward and backward context searches (// and \\).

Regular Expressions

ED supports regular expression notation for specifying patterns in line addresses and in the S, G, and X commands. A regular expression represents one or more strings of characters for which to search. For a description of regular expressions, refer to the chapter on DM basics. The notation is summarized below for your convenience. These search and substitute operations are identical in function to their Display Manager counterparts, although the syntax of the DM's S command differs slightly.

Summary of Regular Expression Notation

```
Literal character
         Any character (except newline)
%
         Beginning of line
         End of line
[...]
         Character class (any one of these characters)
         Negated character class (all characters except those in brackets)
[~...]
[c1-c2] Any single character in the range c1 to c2
         Escaped character (e.g., 0%, 0[, 0*)
@c
en
         Newline
et
         Tab character
         Closure (zero or more occurrences of previous pattern)
{...}
         Tagged pattern
```

ED COMMANDS

The following is a list of ED commands. Default line addresses are in parentheses. Commands may be typed in either upper- or lowercase.

(.)A [text]

The append command reads the text and appends it after the addressed line. The current line is left on the last line input, if any. If no lines are input, the current line remains on the addressed line. Signify the end of the text by typing a line with a period as its first and only character.

(.)B[+/./-][screensize]

The browse command is a shorthand command to print out a screen of data. It has three basic forms, any of which may be followed by a screensize. A simple B (or B+) prints the current line and the screen after it. B- prints the screen of text preceding (and including) the addressed line. B. prints a screen of text, centered on the addressed line. Except for the B. command, these commands leave the current line at the last line printed. The default screensize is 23 lines. If you specify a screensize, it becomes the default screensize for the rest of the editing session or until changed.

(.,.)C [text]

The change command deletes the addressed lines, then accepts input text which replaces these lines. The current line is left at the last line input, if there were any, otherwise at the first line not deleted. Signify the end of the text by typing a line with a period as its first and only character.

(.,.)D The delete command deletes the addressed lines from the buffer. The line originally after the last line deleted becomes the current line; however, if the lines deleted were at the end of the file, the new last line becomes the current line.

E [filename]

The edit command deletes the entire contents of the buffer and then reads in the named file. When it executes this command, ED sets the current line to the last line of the buffer and displays the number of lines read. Also, it remembers the supplied filename for possible use as a default filename in subsequent R or W commands.

F [filename]

If you specify a filename, the currently remembered filename is changed to that name. Otherwise, ED prints the currently remembered filename.

(1,\$)G/regular expression/command

The global command executes the other specified command for every line that matches the regular expression. To execute multiple commands on the lines matched, place each on a separate line and terminate each command except the last with an "at" sign (@). For example,

(.)I <text>

The insert command inserts <text> before the addressed line. The current line becomes the last line input, or, if there are no new lines, the addressed line. This command differs from the A command only in the placement of text. Signify the end of the text by typing a line with a period as its first and only character.

(.,.)K<address>

The kopy command copies the addressed lines to the position after the line specified by <address>. The last of the copied lines becomes the current line.

(.,.)M < address >

The move command deletes the addressed lines from their original location, and places them after the line specified by <address>. The last of the moved lines becomes the current line.

- (.,.)P The print command prints the addressed lines. The last line printed becomes the current line. The P command can be used as a modifier following any other command, except the A, C, I, or Q commands. When used in this way, it prints the last line affected by the command.
- Q The quit command causes ED to exit. If you have not written the file since changing it, ED reminds you once to do so.

(.)R [filename]

The read command reads the named file into the buffer after the addressed line. If you do not specify a filename, ED uses the remembered filename (see E and F commands). The remembered filename is not changed. The address 0 (zero) causes ED to read the file in at the beginning of the buffer. If the read is successful, the number of lines read is displayed. The last line read becomes the current line.

(.,.)S/regular expression/replacement/

(.,.)S/regular expression/replacement/G

The substitute command searches each addressed line for an occurrence of the regular expression. On each line that contains a match, ED replaces the first occurrence of the expression with the replacement string. If the global replacement indicator G follows the command, all occurrences of the regular expression are replaced. The delimiting character for the regular expression and replacement need not be a slash (/); you can use any character except a space or newline. If the substitution fails on all addressed lines, ED prints a question mark (?). The last line substituted becomes the current line. If no regular expression is specified (for example, S//pat/), ED uses the previous regular expression.

An ampersand (&) in the replacement is replaced by the string that matched the regular expression. To suppress this special meaning of &, precede it with an at sign (@). Note that except for &, all characters in the replacement string are inserted literally.

To split or merge lines, use the symbol @n to stand for the NEWLINE character at the end of a line.

(1,\$)W [filename]

The write command writes the addressed lines into the file. If the specified file does not exist, it is created. This command does not change the remembered filename. If no

filename is given, the remembered filename is used (see the E and F commands). The current line is left unchanged. If the command is successful, ED displays the number of lines written.

(1,\$)X/regular expression/command

The except command is the same as the global command except that commands are executed for every line that does not contain a match for the regular expression.

- (.)= The equals command displays the line number of the addressed line. The current line is not changed.
- # comment

Text following a pound sign (#) in the first column of a line is treated as a comment and is ignored by the editor. This command allows ED scripts to contain comments. Only whole-line comments are permitted (i.e., you can't place comments at the end of other legal command lines).

(.+1)<carriage return>

An address alone on a line causes the addressed line to be displayed. A blank line alone is equivalent to '.+1' and thus is useful for stepping through text.

DIAGNOSTICS

file size exceeded

This message is printed whenever the file exceeds 6400 lines.

A message is printed if you attempt to quit without writing a file that you edited. ED requires you to retype the command as a verification.

The error message "?" is printed whenever a command fails or is not understood.

EDACCT (EDIT_ACCT_FILE) -- Create, edit, or list accounts.

FORMAT

EDACCT [options]

EDACCT is used to define accounts in the network registry. Valid person, project, and organization names must have been previously defined with EDPPO.

While all of the EDACCT options are described below, it is unlikely that you will be able to manipulate the network registry unless you are the network administrator for your network. The registry is protected by ACL restrictions. However, you will be able to list registry entries.

For complete information on the use of network administration commands such as EDACCT, see Administering Your DOMAIN System.

OPTIONS

At least one of the following options must be specified; however, you may only include one -A, -D, -C, or -L option per command line. If the command line does NOT include -A, -D, -C, or -L, EDACCT enters an interactive editing session and accepts commands from standard input. See the "EDACCT COMMANDS" section below.

-R pathname

Specify pathname of registry you want to use. You should only use this option with -LOC (described below) to manipulate a remote node's local registry. If you want to manipulate the master registry, omit this option and let EDACCT use the network registry file copy ('/registry/registry') on the current node to locate the master registry.

-A pers proj org homedir password

Add a new account with the person, project, organization, log in home directory, and password indicated. You must specify all fields; you can use "None" to specify proj and/or org. A space between two quotation marks specifies a blank password. This option is not valid for local registries.

-D pers proj org

Delete the account with the person, project, and organization names indicated. You must specify all fields. This option is not valid for local registries.

-C pers proj org [-p passw] [-h homedir]

Change the password and/or the login home directory for the account named. A space between two quotes may be used to specify a blank password. This option is not valid for local registries.

-L [pers [proj [org]]]

List specified entries. All entries for a particular category are listed if you omit that portion of the account specification. You

may also use "%" in any of the fields to match all entries in that field. If you indicate no accounts by name, all accounts are listed.

-LOC

Use the registry '/registry/local_registry' on the node given by the -R option or this node. You may specify only the node name (//node) for "pathname" on -R in this case. If you specify -LOC, the only valid operation is to list entries (-L).

-NP

Suppress prompts for interactive editing.

EXAMPLES

\$ edacct -loc -l List entries in the local account file.

Note that two have expired.

paul	mfg	none	30	83/03/12.12:16	exp:83/03/12	//pj/paul
joe	none	r_d	1FB	83/03/15.08:28	exp:83/03/15	
csa	none	none	30	83/03/15.08:32	exp:83/03/15	//my/csa
rjm	mktg	pay	B8	83/03/15.09:50	exp:83/03/15	//slash/rjm
zoo	cage	feed		83/03/14.20:53	exp:83/03/14	*INVALID* //me
flip	none	wilt	124	83/03/09.18:46	exp:83/03/09	//go/flip
user	none	none	30	83/03/14.19:22	exp:83/03/14	*INVALID* /

EDACCT COMMANDS

The following interactive editing commands may be entered from standard input if you do not include -A, -D, -C, or -L on the EDACCT command line. In all cases, periods may be used instead of blanks to separate person, project, and organization identifiers.

a [pers [proj [org [homedir [password]]]]]

Add a new account. The account is added AFTER the current account (last one listed) unless you have positioned to the top (t). You will be prompted for any fields omitted. Press <RETURN> to assign the default, if appropriate. A space between two quotes may be used to specify a blank password.

d [pers [proj [org]]]

Delete one or more accounts. If no fields are given, only the current entry (last one listed) is deleted. % may be specified for wildcard match, in which case all matching entries (from the top) are deleted. If proj and/or org are omitted, % is assumed for them. Verification of each deletion is requested. The last deleted entry is saved and may be moved to a different place using 'i'.

- i Insert the last deleted entry at the current location (after the current entry or at the top).
- t Move to the top of the file, before the first entry.
- b Move to the bottom of the file, at the last entry.

n [[-]number]

Move up (-) or down "number" entries. For example, "n 2" moves down two entries, while "n -4" moves up four entries. If "number" is omitted, the default is 1.

c [pers [proj [org]]]

Change one or more entries. If no arguments are specified, only the current entry is changed. % specifies wildcard match. If proj and/or org are omitted, % is assumed for them. If arguments are given, the search begins at the top of the file. Wildcarded fields are not eligible for changing. You will be prompted for each new field in order. Press <RETURN> to keep a field unchanged.

l [pers [proj [org]]]

List the current entry (if no arguments are specified) or the first matching entry, starting at the top. % may be used to specify wildcard match.

la [pers [proj [org]]]

List all matching entries, starting at the top. % may be used to specify wildcard match.

ln [pers [proj [org]]]

List the next matching entry, starting at the current entry. If no arguments are given, the previous name specification is used.

lc List all the changed or new entries.

ll List the locality of the current entry, displaying the preceding and succeeding five entries.

wr Update the file with all changes and exit.

q Quit without updating.

h [comm] Help -- list briefly all the available commands or a particular command in detail.

EDACL (EDIT_ACCESS_CONTROL_LIST) -- Edit or list an ACL.

FORMAT

EDACL [commands] [options] pathname...

Every directory and file has an associated access control list (ACL) that lists users and their rights to the object. EDACL edits or displays the ACL of the object(s) specified. The structure and usage of an ACL is described following the examples below.

ARGUMENTS

pathname

(required)

Specify the object whose ACL you wish to edit or display.

Multiple pathnames and wildcarding are permitted.

commands

(optional)

Specify the action(s) described below. If you do not specify a

command, EDACL enters an interactive editing mode.

Default if omitted: read commands from standard input; do not

precede commands with a hyphen (-) in this

mode.

COMMANDS

-L

List ACL entries.

-A ppon rights

Add the specified entry to an ACL. You will receive an error

message if the ACL entry exists.

-AF ppon rights

Add force. Add the specified entry to an ACL. You will not

receive an error message if the ACL entry exists.

-AR ppon rights

Add the specified rights to an ACL. You will receive an error

message if the entry does not exist.

-C ppon rights

Change the access rights in the entry for ppon (replaces current

rights). You will receive an error message if the entry does not

exist.

-CF ppon rights

Change force. Change the access rights in the entry for ppon

(replaces current rights). You will not receive an error message

if the entry does not exist.

-D ppon

Delete the ACL entry for ppon. You will receive an error

message if the entry does not exist. If the 'ppon' is '%.%.%.%',

then EDACL will leave the entry with 'S' and 'E' rights to maintain DOMIAN/IX compatibility.

-DF ppon rights

Delete force. Delete the specified rights from the entry for ppon. You will not receive an error message if the ACL entry does not exist.

-DR ppon rights

Delete the specified rights from the entry for ppon. You will receive an error message if the entry does not exist.

-CDN node

Change the default node ID.

-CN ppon node

Change the node ID entry in ppon.

-Q

Quit without changing the object's ACL. This command is useful only when you supply EDACL commands interactively (see -I).

OPTIONS

-DIR Only operate on directories.

-FILE Only operate on files.

-ID Edit the default initial ACL for directories (-DIR implied).

-IF Edit the default initial ACL for files (-DIR implied).

-UNIX Enable editing of 'S' and 'E' rights for directories; disable the

setting of these rights by default.

The following two options apply only when EDACL reads commands from standard input:

-P EDACL interprets commands when it receives an EOF (usually

CTRL/Z). This is the default when you have redirected standard input (i.e., instructed the program to read commands

from a Shell program, here document, file, or pipe).

-I EDACL interprets commands as you enter them. This is the

default when you have not redirected standard input. You may only specify one pathname (with no wildcards) in this mode. EDACL changes a copy of the ACL; the command does not assign a new ACL to an object until it reads an EOF. Thus, EDACL -I does not change an ACL if you terminate the session

with the "Q" command.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

(The DOMAIN System User's Guide also provides detailed examples of applying and manipulating ACLs.)

1. The order of the commands in the following sequence is significant.

```
$ edacl -L sales

%.%.% pgndwrx

ppon is all wildcards (%.%.%.%), so
all users have complete rights

(pgndwrx) to 'sales'.

$ edacl sales -cf dan.% -none

Deny user DAN access to 'sales'.
```

\$ edacl -L sales
DAN.%.%.% ----%.%.%.% pgndwrx
\$

Deny user DAN access to 'sales'.
Other users still have all rights.
Note that the system automatically
places specific entries before
general ones.

\$ edacl sales -a joe -owner \$ edacl -L sales joe.%.%.% pgndwrx dan.%.%.% ------%.%.%.% pgndwrx Add user JOE to the ACL for 'sales' with all rights.

Allow users in the MKTG organization to change file contents, but do not let them assign rights to others (p and g), change the node ID entry (n), or delete the file (d).

\$ edacl sales -c % r \$ edacl -L sales joe.%.%.% pgndwrx dan.%.%.% -------%.%.mktg.% ----wrx %.%.%.% ----r-- Change everyone else's access to read only. Note that the more liberal rights (wrx) assigned to the MKTG organization in the previous line still apply, since specific entries override general ones.

2. The following examples illustrate the effect of the -UNIX option.

```
$ edacl dir
dir
* 1
 %.%.%.%
                                    pgndcalrse
* a jim -none
 jim.%.%.%
                                     ----se
* a ers -r
 ers.%.%.%
                                    ----rse
 jim.%.%.%
 ers.%.%.%
                                    ----rse
 % . % . % . %
                                    pgndcalrse
Now specify -UNIX ...
$ edacl dir -unix
dir
% . % . % . %
                                    pgndcalrse
* a jim -none
 jim.%.%.%
* a ers -r
 ers.%.%.%
 rees.%.%.%
 ers.%.%.%
                                    ----r--
 % . % . % . %
                                    pgndcalrse
```

WHAT IT ALL MEANS

Every object in the system (whether directory or file) has an access control list that defines who may access that object, and in what ways. The ACL is made up of a series of entries (automatically arranged in increasing order of specificity) that consist of two elements: a subject identifier (SID) and a set of rights (see Fig. 4-1). The SID identifies those users to whom the specified set of rights apply.

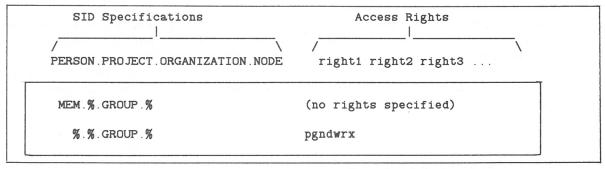


Figure 4-1. An Access Control List (ACL) with Two Entries

The Subject Identifier

The SID is in "ppon" format, i.e.:

PERSON.PROJECT.ORGANIZATION.NODE

The PERSON, PROJECT, and ORGANIZATION identifiers can be any names that exist in the associated network registry files (and that are thus known to the system). The NODE identifier is a hexadecimal node ID number. In addition, you may use the wildcard % in any one of the "ppon" fields. The % wildcard matches zero or more characters -- any name in the network for that field, in other words. You may omit trailing % wildcards and the periods that separate them. For example, the following SIDs correspond to user JOE regardless of his project, organization, or node ID, and are identical:

JOE.%.%.% JOE.% JOE

By convention, users with the project name BACKUP may create backup copies of files and directories on magnetic tape. Users with the project name BACKUP need read (R) access to files and directories. EDACL issues a warning when you change an ACL in a way that denies BACKUP access. However, EDACL does execute the command. Ignore the warning *only* if the object(s) does not require backup copies. If the object(s) does require backup copies, edit the ACL again and grant project BACKUP read access.

Access Rights

You may assign the following access rights to the types of objects indicated:

Files	Directories	Any Object
D Delete R Read W Write X Execute	C Change A Add files and subdirectories L Add links D Delete R Read S Search (SR9.0) E France (SR9.0)	P Protect G Grant N Node
	E Expunge (SR9.0)	

You may specify access rights individually or use abbreviations which stand for certain sets of rights. Table 4-1 defines individual rights. Table 4-2 defines the abbreviations you may use to specify commonly assigned sets of rights.

Note that, starting with SR9.0, there are two special rights ('S' and 'E') that pertain to directory ACLs. These rights have been introduced to support DOMAIN/IX, but the rights are also checked in the AEGIS environment. To preserve compatibility with SR8, all directory ACLs created before SR9.0 have these rights set. Newly created directory ACLs also have these rights set by default. In addition, if the directory ACL does not have a %.%.% entry, the system (by default) adds the entry and grants it only 'S' and 'E' rights.

If you want to manipulate the 'S' and 'E' settings, specify -UNIX when you invoke EDACL. (See Example 2 above.) If you do not want the system to add the default %.%.%.% entry, keep a %.%.%.% entry with no rights (i.e., %.%.%. ------).

We recommend that you leave the 'S' and 'E' rights on, as set by default, since both Apollo and user-supplied software depend on these rights for their proper operation.

ACLs and Directories

In addition to its own ACL, each directory contains within it two additional ACLs (called "initial ACLs"): one for new files and another for new subdirectories that are created within that

Table 4-1. Access Rights for Files and Directories

	.		
Access Right	Abbreviation	Meaning for Directories	Meaning for Files
Protect	P	Change the object's ACL	
Grant	G	Grant any subset of your rights to other users	
Node	N	Change the nodusers may acce	
Delete	D	Delete the directory	Delete the file
Read	R	List entries	Read file contents
Write	W		Write to the file
Execute	x	- 192 - 192	Execute object file
Change	С	Change names and delete links	
Links	L	Add links	
Add	A	Add files and subdirectories	
Search (DOMAIN/IX)	S	Allow directory to be searched for subordinate objects	
Expunge (DOMAIN/IX)	E	Allow subordinate object(s) to be deleted (also need 'D' rights on the objects)	

NOTE: To delete a tree you need directory delete rights, directory expunge rights, directory change rights (if the directory contains links), and file delete rights (if the directory contains files).

Table 4-2. Abbreviations for Commonly Assigned Rights

Term	Meaning	Directories	Files
-OWNER	All rights	PGNDCALRSE	PGNDWRX
-USER	All rights except ability to change ACL	DCALRSE	DWRX
-READ	File read access	Not allowed	R
-EXEC	File read access Execute access to Object files	Not allowed	RX
-LDIR	List directories	RSE	Not allowed
-ADIR	List directories and add entries	ALRSE	Not allowed
-NONE	Grant no rights To provide SR8 com-	SE or None	None
	patibility, S and E rights are set by default. If -UNIX was present on the command line, all directory rights are revoked.		

directory. When you create a new file or directory (either by outright creation or by copying it to a new location in the file hierarchy), the system assigns an ACL to it by copying the appropriate initial ACL stored in the parent directory. In addition, if the newly created object is itself a directory, the two initial ACLs from the parent are replicated in the new subdirectory, unless you specifically indicate otherwise (see the CPT (COPY_TREE) command).

Figure 4-2 should make this more clear.

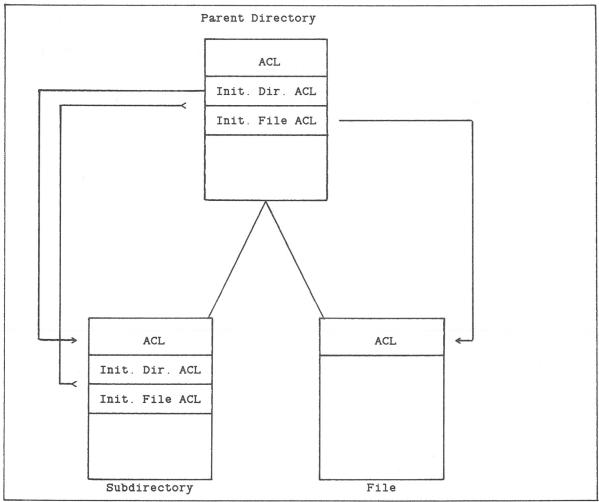


Figure 4-2. Applying Initial ACLs

The various options on the EDACL and ACL commands determine which of these several access control lists you are editing, copying, or displaying.

NOTE: EDACL will not allow an operation that would restrict everyone from changing an ACL. At least one user must have the right to change the ACL (P).

You need N (change node) rights to change an object's node list, or to grant other users N rights.

The -CDN and -CN commands require N (change node) rights. When a user without N rights adds an entry to an ACL, that entry will always receive the default node ID (%), even if the user specifies a different node ID.

EDFONT

EDFONT -- Edit a character font.

FORMAT

EDFONT

EDFONT is an interactive, menu-driven program that allows you to create, edit and view character font files. Its user interface is based on a pointing device (such as a mouse or touchpad), in contrast to the older, function-key menus of OLD_EDFONT. For a detailed explanation on editing a character font, see the EDFONT appendix.

EDFONT requires no arguments or options to begin processing.

EDMTDESC (EDIT_MAGTAPE_DESCRIPTOR) -- Edit magtape descriptor file.

FORMAT

EDMTDESC pathname {options}

EDMTDESC allows you to create, list, and modify magnetic tape descriptor objects. These descriptor files provide information to the streams manager so that it can handle subsequent tape operations much as an SIO descriptor file describes the configuration of an SIO line.

ARGUMENTS

pathname

(required)

Specify name of magtape descriptor file to be created, listed, or edited.

OPTIONS

At least one of the following options must be specified.

-C Create a new magtape descriptor object with the name given in the 'pathname' argument.

-L [var...] List the values of the variable(s) specified. If no variables are named, the entire magtape descriptor is listed.

-S {var value}...

Set the variable(s) indicated to the specified value(s). At least one variable/value pair is required if -S is specified. Multiple variable/value pairs are permitted, separated by blanks.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

VARIABLES

The variables known to EDMTDESC are listed below, along with their types and default values. The variable types are: integer (int), Boolean (y/n), character string of n letters (c [n]), and date (in format yy/mm/dd.hh:mm).

name	type	default	definition
DEV	c[1]	М	device type ('M' for magtape, 'C' for cartridge)
U	int	0	magtape unit number (normally 0)
LAB	y/n	yes	'yes' if magtape is ANSI labeled, 'no' if unlabeled
REO	y/n	no	'yes' to reopen previously used volume, 'no' to open new volume ('yes' suppresses rewind)
CLV	y/n	yes	'yes' closes volume when file is closed, 'no' leaves volume open
SPOS	y/n	no	'yes' saves volume position when volume is closed (for reopen), 'no' rewinds volume when closed
VID	c[6]	-auto	volume identifier (labeled volumes)
VACC	c[1]		volume accessibility (labeled volumes)
OWN	c[14]	-auto	volume owner (labeled volumes)
F	int*	1	file sequence number — integer or "cur" for current file, or "end" for new file at end of labeled volume
RF	c[1]	D	record format "F" for fixed length, "D"
		_	for variable length, "S" for spanned, "U" for undefined
BL	int	2048	block length, in bytes
RL	int	2048	(maximum) record length, in bytes
ASCNL	y/n	yes	'yes' for ascii newline handling (strip
			newlines on write, supply them on read),
			'no' for no newline handling
FSECT	int	1	file section number (labeled volumes)
FID	c[17]		file identifier (labeled volumes)
FSID	c[6]		file set identifier (labeled volumes)
GEN	int	1	generation of file (labeled volumes)
GENV	int	1	generation version of file (labeled volumes)
CDATE	date	-auto	creation date of file (labeled volumes)
EDATE	date	-auto	expiration date of file (labeled volumes)
FACC	c[1]		file accessibility (labeled volumes)
SYSC	c[xx]		system code (labeled volumes)
SYSU	c[xx]		system use (labeled volumes)
BOFF	int	0	buffer offset (labeled volumes, should be 0)

For cartridge tape (DEV C), you must change the block length (BL) and the record length (RL) to be 512 or less and the record format to be fixed ('RF F').

EXAMPLES

\$ edmtdesc ct -c -s dev c bl 512 rl 128 rf f

Create descriptor file ct for cartridge tape, blocking 4 records of maximum length 128 to each block.

EDNS (EDIT_NAMING_SERVER) -- Invoke naming server editor.

FORMAT

EDNS [node_id]

EDNS allows you to inspect and/or modify NS_HELPER's master network root directory and replica list. Once invoked, EDNS enters an interactive mode and accepts the commands described below. For complete information on NS_HELPER and EDNS usage, see Administering Your DOMAIN System.

ARGUMENTS

node_id

(optional)

Set the default NS_HELPER to the NS_HELPER at the node

specified by the hexadecimal node id.

Default if omitted: set the default NS_HELPER to any active

NS_HELPER.

COMMAND SUMMARY

For complete information on EDNS command usage, see Administering Your DOMAIN System.

SYNTAX

(abbreviation shown in UPPER case)

FUNCTION

Default options are indicated by "(D)".

ADD name node_id

Add an entry directory name, its node_id, and its uid to the master root directory; the node must be connected to the network when this command is executed so that the uid of its entry directory may be

retrieved.

REPlace name node id

Update the node_id and uid values for entry 'name'. Use this command after running the utilities CHUVOL or INVOL,

or after replacing a disk.

DELete name

Delete name from the master root

directory.

EDNS (EDIT_NAMING SERVER)

LD [names] [-NODE node_id]

[-sn|-nsn] [-t] [-u] [-n] [-dte] List root directory entries by name; if names are specified, only those names are listed.

List all entries with node_id specified; if 'names' are also specified, those names and entries with 'node_id' are

listed.

-sn

-NODE node id

-nsn

List entries sorted by name.

Suppress name sorting.

The following options specify what special information should be displayed with each entry that is listed:

(D)

-u-n

-dte

Display entry type.

Display uid. Display node id.

Display date/time this entry was made to the directory and the node_id of the replica where this entry was made.

ADDRep node

Add node to replica list.

DELRep node

Delete node from replica list. This command causes the NS HELPER replica at 'node' to delete its database and stop running after its propagation list has emptied. It also causes this replica to be removed from all

other replica lists.

SHUT node

Shut down a replica. This command causes the NS_HELPER replica at 'node' to delete its database and stop running immediately. The shutdown replica is not removed from the other replica lists.

LR [-clocks]

Display list of replicas in the network.

-clocks

Display each replica's current clock date/time and check for any replicas whose clocks are skewed.

MERGE [nodeA] -FROM nodeB

Merge all entries in the root directory and replica list of the NS HELPER at nodeB into the root directory and replica list of the NS HELPER at nodeA. If nodeA is not specified, merge into the default NS_HELPER's directory and replica list.

MERGE ALL [node]

Do a global merge using the NS HELPER at 'node' (or the default NS HELPER if node is not specified) as the base for the merge.

INIT [nodeA] [-FROM nodeB]

Initialize the NS_HELPER at nodeA or, if nodeA is not specified, the default NS HELPER.

If -FROM nodeB is not specified, get a list of all node_ids, their entry directory names, and uids and add them to the NS_HELPER at nodeA's directory. If -FROM nodeB is specified, merge all entries in the directory and replica list of NS_HELPER at nodeB into the NS_HELPER at nodeA's directory and replica list; then add nodeA to nodeB's replica list.

DIFF [nodeA] nodeB

List differences between the NS_HELPER at nodeA's (or the default NS_HELPER's) directory and the NS_HELPER at nodeB's directory (e.g. entries in only one of the directories, names that match whose types or uids are different, etc.)

List differences between the NS_HELPER at nodeA's replica list and the NS_HELPER at nodeB's replica list (e.g. replicas which appear in one NS_HELPER's list but not in the other's).

INFO

Give the node_id and status of the current default NS_HELPER.

SET [nodeA]

Subsequent commands which do not specify an NS_HELPER, will be directed to the NS_HELPER at nodeA. If 'nodeA' is omitted, the default NS_HELPER is set to an active server on the local net.

Quit

Quits your current EDNS session.

NOTE: a 'node' argument may be either a node name or a node_id; if it is a node name, the corresponding node_id will be found by looking up the name in the default NS_HELPER's directory.

EDPPO (EDIT_PPO_FILE) -- Edit/list person, project, or organization names.

FORMAT

EDPPO {options}

EDPPO is used to define usernames in the network registry. This is a prelude to creating network accounts, which associate usernames with login home directories and passwords. Use EDACCT (EDIT_ACCOUNT) to perform that operation.

While all of the available EDPPO options are described below, you will probably not be able to edit network registry files unless you are the network administrator for your network. You should, however, be able to list the contents of the network and local registries. Use EDPPO to find out who is associated with a particular username and account.

For detailed examples of editing network registry files and complete information on network registry commands, see Administering Your DOMAIN System.

OPTIONS

If you omit the -A, -D, -L, and -LF options, EDPPO enters an interactive editing session which reads commands from standard input as described in the COMMANDS section below.

Default options are indicated by "(D)."

-R	กล	th	n	a.m	e
10		· ULL	44	α	

Specify the registry to be edited or listed. You should only use this option with -LOC (described below) to edit or list a remote node's local registry. If you want to edit or list the master registry, omit this option and let EDPPO use the network registry file copy ('/registry/registry') on the current node to locate the master registry.

-LOC

Specify that the node's local registry ('/registry/local_registry') is to be edited or listed. If you specify -LOC, the only other valid options are -R -L, -LF, -COL, and -D. You may not manipulate the local registry using EDPPO's interactive mode; the list or delete functions must be specified on the command line.

-PERS (D)

Edit or list the contents of the Person file.

-PROJ

Edit or list the contents of the Project file.

-ORG

Edit or list the contents of the Organization file.

-A name [fullname]

Add a name and optional full name text. PPO names can be up to 32 characters long, must begin with a letter and can include only letters, numbers, and underscores (_). PPO names are automatically maintained in lowercase. Associated full name

text is optional but strongly recommended. It can include any characters and be up to 32 characters long. Use single (or double) quotation marks to embed spaces (or quotation marks) in a full name.

-D name

Delete a name. Command line delete is valid only on a local registry. To delete a name from the network registry file, use EDPPO's interactive mode. DELETE WITH EXTREME CAUTION, even locally, because once you delete a name no one can EVER access any files created by the user who had that PPO. (The system cannot recreate a unique name identifier once it has been deleted.) Normally, you should delete people's accounts (using EDACCT), NOT their names.

-L [name...]

List the name(s) specified. If 'name' is omitted, all names are

listed

-LF [name ...]

List the name(s) specified, along with associated full name text,

if any. If 'name' is omitted, all names are listed.

-COL

List the names in a single column.

-NP

Suppress prompts during interactive editing.

COMMANDS

If you omit the -A, -D, -L, and -LF options on the command line, EDPPO enters an interactive editing session which accepts the following commands from standard input.

a [name [fullname]]

Define one or more new names, along with optional full name. If 'fullname' is omitted, EDPPO will prompt you for it. Enter an empty line if you wish the fullname field to be null. (See -A above for name formats and restrictions.) If no name is specified, names are read from standard input. We strongly recommend that you supply full names.

cf name new_fullname

Change the full name associated with the name indicated.

d name

Delete the name indicated. Once deleted, the unique name identifier cannot be recreated, so DELETE WITH EXTREME CAUTION. Once you delete a name, no one can EVER access any files created by the user who had that PPO. Normally, you should delete people's accounts (using EDACCT), NOT their

names.

l [name]

List one or all names defined (including new ones).

lf [name]

List one or all names defined with their associated full names, if

any.

ln

List only those names added during this editing session.

wr

Update the file with the changes and exit.

EDPPO (EDIT_PPO_FILE)

q

Quit without updating.

h [comm]

Help -- list briefly all the available commands or a particular command in detail.

EXAMPLES

List person names in the local registry.

\$ edppo -loc	edppo -loc -l						
adm	alan	apgar	beth	bls			
bso	burt	cas	charlie	chris			
chrissy	cmt	col	color	csa			
sqh	sys_person	taylor	todd	user			
vic	wilson j	zahn					

EDSTR (EDIT_STREAM) -- Edit a stream.

FORMAT

EDSTR { command | -E command | -F cmdfile ...} [pathname] [option]

EDSTR copies the named input files to standard output, performing editing as directed by EDSTR commands in the command line or in the named command file.

ARGUMENTS

If neither the -E or -F argument is specified, EDSTR assumes that the first token on the command line without a hyphen is an EDSTR command (see below) and that the remaining tokens (if any) are pathnames.

command

(optional)

Specify a single EDSTR command (except A, C, or I). EDSTR accepts the ED commands A, C, D, I, P, R, S, W, and =. To use the A, C, or I commands, place them in a command file as described below.

Default if omitted: use -E and/or -F

The following two arguments may be repeated and intermixed in any order. EDSTR executes them in the order in which they appear on the command line.

-E command

(optional)

Specify an EDSTR command (except A, C, or I). To use the A, C, or I commands, place them in a command file as described below. EDSTR can accommodate commands totaling approximately 5000 characters (including <text> arguments), and lines up to 120 characters long.

Default if omitted: use 'command' or -F

-F cmdfile

(optional)

Specify a file containing EDSTR commands, one per line. Control is passed to this file for command processing. See -E for EDSTR command restrictions.

Default if omitted: use 'command' or -E

pathname

(optional)

Specify input file to be edited. Multiple pathnames are

permitted.

Default if omitted: edit standard input

OPTIONS

-N

Supress writing of output except for P and W EDSTR commands. By default, EDSTR writes each line of input to standard output after editing.

EXAMPLES

\$ edstr -e s/joe/mary/g -e 20r add_stuff infile >outfile

This command first replaces all occurrences of "joe" with "mary", then copies material in the file "add_stuff" into "infile" following line 20. Results are written to the file "outfile".

SUMMARY OF EDSTR COMMANDS

Addresses:

17 a decimal number

\$ the last line of the file

/pat/ search forward for line containing pat

pat\ search backward for line containing pat

line+n n lines forward from line

line-n n lines backward from line

Defaults:

- () (no address) use current line (+1) use the next line (1,\$) use all lines
- Commands:

()	A	Append text after line (text follows)
()	C	Change text (text follows)
()	D	Delete text
()	I	Insert text before line (text follows)
()	P	Print text (can be appended to other commands)
()	R file	Read file, appending after line
()	S/pat/new/GP	Substitute new for leftmost pat (G implies all occurrences)
(1,\$)	W file	Write file, leave current text unaltered (if
()	=[P]	no file is specified, write to current filename) Print line number, current line

Arguments:

\$n Write to/read from the nth temporary buffer

EM3270 (EMULATE 3270) -- Emulate an IBM 3270 terminal.

FORMAT

EM3270.{device}

EM3270 allows a DOMAIN node to emulate an IBM 3270 terminal over an SIO line connected to a VT100-to-3270 converter. The command is meaningless without this additional hardware.

While EM3270 requires no arguments or options, there are actually three different commands, depending on which protocol converter you are using. The following protocol converters support the EM3270 Package software:

- ICCI Model CA20
- ICCI Model CA12
- KMW Model BAC-3270 FS
- PCI 1076

Specify the device name with the EM3270 command. For example:

\$ em3270.pci

if you are using the PCI 1076 protocol converter.

Follow the manufacturer's directions for connecting the converter you choose to the node's SIO lines.

EM3270 COMMANDS

Once you have invoked EM3270, you may use the following commands:

H Display command summary information.

LI [n] Select SIO line n. The default SIO line is 1.

Q Exit from EM3270.

SPEED n Set SIO line speed. Valid speeds are 50, 75, 110, 134, 150, 300, 600,

1200, 2000, 2400, 3600, 4800, 7200, 9600, and 19200.

[NO]SYNC Enable/disable XON/XOFF on the SIO line.

In addition to these commands, two control/key sequences perform special functions:

CTRL/<F8> Switch between command mode and Remote 3270 mode.

CTRL/<F7> Display a layout of the 3270 emulation keyboard.

EMT (EMULATE_TERMINAL) -- Emulate a dumb terminal.

FORMAT

EMT [pathname]

EMT allows your node to emulate an ASCII terminal connected to another computer. This asynchronous connection exists through a stream opened on one of the node's SIO lines. EMT also permits ASCII file transfer between your node and the remote host.

ARGUMENTS

pathname

(optional)

Specify file containing EMT commands.

Default if omitted: read commands from standard input

USING EMT

EMT begins execution in local mode, and displays the following prompt:

emt>

To enter remote mode, press <F1>. (The EMT command DL no longer exists.) In remote mode, your terminal operates as if it is physically connected to the remote computer ("host"). You can log on and enter remote host commands.

To return to local mode, press <F1> again.

Input/Output Streams

EMT uses the four standard streams (standard input, standard output, error input, and error output) as follows:

- EMT commands are read from an EMT command file or from standard input. The command filename may be specified on the command line or using the EMT 'run' command. Up to four levels of command files may be nested. When EOF is reached on a command file, commands are read from the previous file or from standard input. If EOF is reached on standard input, EMT exits.
- Keystrokes to be sent to the host computer are read from error input. Error input may not be redirected to a file. Use the EMT 'xmit' command to transmit a file (of commands or data) to the host. Use the EMT 'rcv' command to receive host transmissions to a DOMAIN file.
- EMT Command responses and all messages from the host are written to standard output.
- Error messages from AEGIS system calls are written to error output. Optional monitoring (MONIT) may also be written to error output (or to a named file).

Transferring Files

You can transfer files using EMT's receive (RCV) or transmit (XMIT) commands. XMIT sends a DOMAIN file to the remote host. RCV opens a DOMAIN file to receive information from the remote host. For example, if you type (in local mode):

emt> XMIT FILEA

EMT displays the following message:

Ready to transmit file FILEA

Next, press <F1>. EMT enters remote mode, and transmits FILEA to the remote host.

If you type:

emt> RCV FILEB

EMT displays this message:

Ready to receive file FILEB.

Next, enter remote mode by pressing <F1>. Use a remote host command to display the information that you want FILEB to receive. EMT automatically writes this and all subsequent host transmissions into FILEB. To stop the RCV, press <F2>.

Transmission Conventions

Use the EMT command INTERM to specify the line terminator used by the host. If you do not know what the host uses as a line terminator, experiment by changing INTERM. Use the EMT command OUTTERM to specify the line terminator to be transmitted to the host.

EMT allows you to open only one DOMAIN file at a time. If EMT receives a XMIT or RCV command while another DOMAIN file is active, it closes the open DOMAIN file, and executes the new command.

During remote mode, EMT waits on both the keyboard and SIO line for characters to process, and monitors the data for characters of special interest to EMT.

You can specify which keyboard characters EMT should interpret by placing the keyboard in raw or cooked mode. In raw mode, EMT passes all keyboard input (except the function keys, keys L1 through L12, and keys R1 through R4), directly to the host. Cooked mode lets you use many of the Display Manager's features for editing the input pad. EMT places your keyboard in cooked mode by default.

EMT COMMANDS

Keys

<F1> Switch between local and remote modes.

<F2> Interrupt a file transfer and close the file.

<F3> Turn TEE on or off. TEE on causes EMT to display file transmission records on the screen. You can use this feature to monitor file transfers, and decide if and when you should stop or interrupt a transfer. The default is TEE on.

<F8> Send a BREAK to the host.

CTRL/<F7> Display function key definitions.

These function keys may be simulated by typing the EMTESC character followed by the function key number (i.e., ~ 1 for F1). When EMT is used from the VT100 emulator, F1-shifted is used instead of F2, and F1-control is used instead of F3.

Commands

AE Abort on error.

ASConly | NOTASConly

Sift out most non-printing ASCII codes. Eliminates triangles, allows BS, CR, ESC, FF, LF, TAB. The default is NOTASC.

BREAK [n] Set the BREAK duration value to n milliseconds. The default is 200. If set to 0, the <F8> (break) key does nothing.

CLOSE Deactivate an RCV file. See the RCV command for related information.

CODE [xx | NONE]

Set the HOST-COMMAND-CODE to the hexadecimal number xx. The default is NONE.

Place the keyboard in cooked mode. This enables many Display Manager features for editing the input pad, and provides an escape sequence for sending control characters to the remote host. To send the host a CTRL character, precede the character with a tilde (~). The sequence ~_ transmits a delete character. To send the host a single tilde character, type ~~.

The EMT default is cooked mode. Cooked mode always echos keystrokes, so it does not require a full duplex connection to the host. (See the RAW c mmand for related information.) Note that the COOKED and RAW commands refer only to the transcript pad and keyboard input. The SIO line itself is always in RAW mode.

EMTESC [chr|NONE]

Set the EMT escape character to chr. Use NONE to disable the escape character. Default is ~ for cooked mode, NONE for raw mode.

The following three commands are useful when standard input is redirected to a file of EMT commands:

F1 Enter remote mode (Simulate Function key F1).

F2 Terminate file transfer (Simulate Function key F2).

F3 Toggle TEE mode (Simulate Function key F3).

HANGUP Cause modem to break connection with the remote host.

HELP [tctl] Display information about EMT commands or about TCTL

commands.

LINE {1|2|3|pathname}

Select the SIO line. Pathname must specify an SIO device descriptor (e.g., /DEV/SIO2). The default SIO line is 1 (/DEV/SIO1).

Display the current SIO line, all EMT switch settings and the receive

filename, if any.

MONIT [pathname]

L

Write every character received over the SIO line to 'pathname'. If a filename is not specified, the previous specification or error output is

used.

NOMONIT Stop monitoring.

QUIT End the EMT session.

RAW [-ECHO]-NOECHO] [-LF]-NOLF]

Place the keyboard in raw mode. This sends keyboard input directly to the remote host, interpreting only function keys. The -ECHO option echos keystrokes on standard output; you should use it when the host is in half-duplex mode. The default is -NOECHO. The -LF option converts CR to LF for lines echoed. The default is -NOLF. (See the COOKED command for related information.) Note: The -ECHO and -LF options are purely local functions that enable you to read what you type. They do not in any way change host/node transmissions.

RCV [-R] [-KEYS|-NOKEYS] [pathname]

Prepare the DOMAIN file specified to receive remote host transmissions. If 'pathname' already exists, EMT appends the transmission to it, unless you specify -R. The receive begins when you enter remote mode <F1>. If you omit 'pathname', EMT uses the previous name, if any. The -KEYS option writes keystrokes to the file along with received data. The default is -NOKEYS.

EMT allows you to interrupt an RCV command at any time by pressing <F2>. EMT remains in whatever mode it was in, but keeps the RCV file active. When you are ready to continue receiving host transmissions, you may type RCV again (in local mode) without a filename, and EMT will use the same RCV file.

If you omit filename and no RCV file is active, EMT issues an error message. If you specify a new RCV file while another RCV file is active, RCV closes the active file, and prepares the new file to receive the transmission.

Use the CLOSE command to deactivate an RCV file.

TCTL {tctl commands}

Pass this command line to the Shell command TCTL to configure the SIO line. It is not necessary to specify the line number (-line), although you may if you wish to operate on a different line than the one you are using. The SPEED and SYNC commands have been superseded by this direct invocation of TCTL.

INTERM {CR|LF|CRLF|VAX|'hex'}

Select the input line terminator. The default is CRLF.

OUTTERM {CR|LF|CRLF|'hex'}

Select the output line terminator. The default is CR. EMT transmits the selected hexadecimal value as the terminator for each line.

XMIT pathname Prepare to transmit the DOMAIN file specified to the remote host. If you omit 'pathname', or if you specify a file that does not exist, EMT issues an error message. When you issue this command, EMT remains in local mode. EMT transmits the file when you press <F1>.

When EMT completes the transfer, it closes the file and returns to the previous mode. EMT does not send an end-of-file (EOF) signal to the remote host. If the host requires an EOF, enter remote mode and transmit it manually.

EMT can also receive commands from the host. If the host transmits the sequence:

HOST-COMMAND-CODE (EMT COMMAND STRING) LINE-TERMINATOR

EMT interprets the string as an EMT command. Use the EMT command CODE to define HOST-COMMAND-CODE.

Input Line Terminators	EMT Response
CRLF	Converts sequence to a line feed, ignoring any null characters that may separate the pair.
CR	Converts sequence to a line feed and ignores LFs.
LF	Interprets it as a line feed, and ignores CRs.
VAX	Interprets both CR and CR-LF as terminators and converts them to line feed.
'hex'	Converts the given hexadecimal value to LF.

ENSUBS (ENTER_SUBSYSTEM) -- Enter a protected subsystem.

FORMAT

ENSUBS subsystem_name

ENSUBS is used to enter a protected subsystem at Shell command level. ENSUBS creates a new process in which to run the subsystem Shell.

Once in the subsystem, the SUBS command can be used to create new managers for the subsystem or to seal data objects so that only managers of the subsystem can operate on them. Also, subsystem managers can be debugged conveniently in this mode using DEBUG, and protected data objects can be examined. Note, however, that access to protected objects requires prior use of the SUBS -UP command.

ARGUMENTS

subsystem_name

(required)

Specify name of subsystem to be entered. The Shell will search the directory /SYS/SUBSYS for the file specified.

NOTES

• ENSUBS XXX just invokes the command /SYS/SUBSYS/XXX in a new process (unless the current process is already running in Subsystem XXX). The new process shares the window of the creating process, and is therefore subject to the same restrictions found when logging into a window. To avoid the limitations, a new window, containing a Shell running in the subsystem, can be created using the DM. Press the <CMD> key; next to the prompt, type:

Command: cp /sys/subsys/xxx

• The access control list on the file /SYS/SUBSYS/subsystem_name determines who can enter the subsystem 'subsystem_name': whoever has read and execute rights to it can enter the subsystem. Usually, this capability should be restricted to the creators of the subsystem or to the System Administrator.

EOFF -- Deactivate the Shell's -E flag.

FORMAT

EOFF

EOFF disables variable evaluation. Variables are evaluated only inside variable expression delimeters, ((expression)); otherwise, the Shell treats the ^var_name expressions as strings and they are not evaluated. To enable variable evaluation regardless of the context in which the variable appears, specify EON.

By default, EOFF is in effect when a Shell is invoked.

If EOFF is specified in a Shell script, it remains in effect until that Shell script exits, or until over-ridden by an EON in a nested Shell script. When a Shell script exits, the state of variable evaluation is returned to the state in effect just before the script was invoked.

EOFF requires no arguments or options.

EON -- Activate the Shell's -E flag.

FORMAT

EON

EON enables variable evaluation regardless of the context in which the variables appear. Normally, variables are evaluated only inside variable expression delimiters, ((expression)); otherwise, the Shell treats the `var_name expressions as strings and they are not evaluated.

By default, EOFF is in effect when a Shell is invoked.

If EON is turned on in a Shell script, it remains on until that Shell script exits, or until over-ridden by an EOFF in a nested Shell script. When a Shell script exits, the state of variable evaluation is returned to the state in effect just before the script was invoked.

EON requires no arguments or options.

EQS (EQUAL_STRING) -- Compare strings for equality.

FORMAT

EQS [string1 [string2]]

EQS compares strings for equality, and sets the abort severity level accordingly.

ARGUMENTS

If no arguments are specified, EQS always returns TRUE.

string1

(optional)

Specify text string to test. If this is the only string given (i.e., 'string2' is not specified), return TRUE if 'string1' is empty;

otherwise return FALSE.

Default if omitted: return TRUE

string2

(optional)

Specify text string to compare against 'string1'. EQS returns

TRUE if the strings are equal, and FALSE if they are not.

Default if omitted: test 'string1' only

EXAMPLES

The following Shell script will compile the PASCAL module named by the first argument (1) if the second argument (2) is '-c'. Then it will bind the module with 'library'.

if eqs ^2 '-c' then pas ^1 endif bind ^1.bin library -b ^1 $\,$

If the second argument is not '-c', or if there is no second argument, the program simply binds the module.

ESA (EXTERNAL_SYMBOL_ADDRESS) -- Display address of external symbol.

FORMAT

ESA symbol_name

ESA displays the address of an external symbol in an installed library. This command is primarily intended for system-level debugging.

ARGUMENTS

symbol_name

(required)

Specify symbol whose address you wish to display. ESA is case sensitive with respect to the symbol name. Lowercase must be used to refer to symbols defined in FORTRAN and Pascal programs. Mixed case may be used, as needed, for symbols defined in C programs.

EXAMPLES

\$ esa stream_\$get_rec
068186
\$

The command above displays the address of STREAM_\$GET_REC. This symbol resides within the streams library, which was installed at system startup time.

EXFLD (EXTRACT_FIELDS) -- Manipulate fields of data.

FORMAT

EXFLD {field_spec} output_format [pathname ...]

EXFLD manipulates data kept in formatted fields. It copies data from specified fields of the input files to specified places in standard output.

ARGUMENTS

field _ spec
(required)

Specify either one of the following two arguments:

field_list

Integer list identifying fields in the input file to be copied. Up to 9 input fields are allowed. You can specify a field by the columns in which it occurs or by its starting column and length. For example, 5-10 denotes a field that extends from column 5 through column 10, and 3+2 denotes a field that starts in column 3 and spans 2 columns. When specifying more than one field, separate the specifications with commas, for example:

5-10,16,72+8

Fields can overlap, and need not be in ascending numerical order. Thus

1-25,10,3

is a valid field specification.

-T [c]

Free-format separator specification. If input fields do not fall in certain columns, but rather are separated by some character (such as a blank or a comma), describe the fields by using '-T c', replacing 'c' with the appropriate separator. A tab character is the default for 'c'.

output_format
(required)

Specify literal string representing output format. Fields from input are referred to as \$1, \$2, \$3, and so forth, denoting the order in which the fields are specified. Up to 9 fields are allowed, plus the argument \$0 which refers to the whole line. Place the \$n symbol in the output format wherever the corresponding field should appear, surrounded by any characters desired. For example, an output format of:

"\$2 somewords \$1"

would produce an output line such as:

EXFLD (EXTRACT_FIELDS)

field2 somewords field1

pathname

(optional)

Specify input file to be manipulated.

Default if omitted: read standard input

EXAMPLES

\$ exfld 1-5,14-18 "\$2 follows \$1"
ABCDE is not DEFGH
DEFGH follows ABCDE
*** EOF ***

Specify extraction.
Input text from standard input.
Result.
Signal completion with CTRL/Z.

EXISTF -- Check for existence of an object.

FORMAT

EXISTF pathname ...

EXISTF reads the object pathname(s) you supply and checks to see if the object exists. If the object does exist, EXISTF returns with a good program status (PGM_\$TRUE). If the object does not exist, EXISTF returns an error status (PGM_\$FALSE).

ARGUMENTS

pathname

(required)

Specify the object to be checked. Multiple pathnames and wildcarding are permitted. If you specify more than one pathname, all the objects must exist for EXISTF to return TRUE.

EXAMPLES

1. \$ if existf my_file then args "The file is there." Test for "my_file"
 \$_else args "Out of luck." endif
 Out of luck.
 exist.

EXISTVAR (EXIST_VARIABLE)

EXISTVAR (EXIST_VARIABLE) -- Check that a variable is set.

FORMAT

EXISTVAR var_name ...

The EXISTVAR command checks to see if the variable name(s) declared as its argument(s) has a currently set value. If the variable is currently set, EXISTVAR returns a "TRUE" value. If the variable is not currently set, EXISTVAR returns "FALSE". If you specify more than one variable name to check, all the variables must exist for EXISTVAR to return "TRUE".

ARGUMENTS

var_name [...]
(required)

Specify the variable name to be checked. Multiple names are permitted, separated by blanks.

EXIT -- Exit from a loop.

FORMAT

EXIT

EXIT terminates the flow of control in a Shell loop construct (FOR, SELECT, and WHILE). When EXIT is encountered, control passes to the first command following the body of the loop (see example below).

You may also interrupt the flow of control in a loop without actually leaving the loop by using the NEXT command. See the NEXT command description for more information.

Do not confuse this command with the DM command EX, which exits the Display Manger and returns control to the Boot Shell. See the EX command description in the DM commands chapter for more information.

The EXIT command requires no arguments or options.

EXAMPLES

Consider the following section from a Shell script:

```
WHILE ((true))

DO READC a

IF ((^a = "y")) THEN EXIT ENDIF

ARGS "still looking ..."

ENDDO

ARGS "Finished."
```

When the READC (READ_CHARACTER) command reads a character into variable "a" that matches the character "y", the EXIT command executes and causes the script to jump to the command following the ENDDO.

For more information on variables, refer to the DOMAIN System User's Guide.

EXPORT -- Change a Shell variable into an Environment variable.

FORMAT

EXPORT var_name...

The Shell can access environment variables using all of the standard variable commands and operators. The EXPORT command adds the capability of turning regular Shell variables into environment variables.

Environment variables are variables that programs can access or set and that are used to store global state information. Several are generated automatically when you create a process; they can be displayed using the LVAR (LIST_VARIABLES) command. For example:

```
$ lvar
environment NODETYPE = DN400
environment TZ = EST5EDT
environment PATH = :~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo_15P
environment HOME = //node_8e4/joseph
environment USER = joseph
environment LOGNAME = joseph
environment PROJECT = none
environment ORGANIZATION = r_d
environment NODEID = 8E4
```

Environment variables are of special interest to users of DOMAIN/IX. Consult the DOMAIN/IX documentation for additional information.

NOTE: The Shell creates environment variables in UPPERCASE only. (Environment variables are case sensitive in DOMAIN/IX; the Shell only allows uppercase ones to avoid collisions between environment variables and Shell variables.)

ARGUMENTS

var_name

(required)

Specify the Shell variable to be changed into an environment variable. It doesn't matter whether or not the name is typed in uppercase; the Shell converts it to uppercase automatically. Multiple variable names are permitted, separated by blanks. If the specified variable does not exist, EXPORT creates it.

EXAMPLES

```
$ eon
$ current_dir := "//panacea/joe"
$ lvar
string current_dir = //panacea/joe (Shell variable created.)
environment USER = joe
environment LOGNAME = joe
```

```
environment PROJECT = none
environment ORGANIZATION = r d
environment NODEID = D5B
environment PATH = :~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo 19L
environment NODETYPE = DN300
environment TZ = EST5EDT
environment HOME = //panacea/joe
$ export current_dir
$ lvar
environment USER = joe
environment LOGNAME = joe
environment PROJECT = none
environment ORGANIZATION = r d
environment NODEID = D5B
environment PATH = :~com:/usr/ucb:/bin:/com:/usr/bin
environment TERM = apollo_19L
environment NODETYPE = DN300
environment TZ = EST5EDT
environment HOME = //panacea/joe
environment CURRENT DIR = //panacea/joe (Environment variable created.)
```

FIND_ORPHANS -- Locate and catalog uncataloged objects.

FORMAT

FIND_ORPHANS [options] [volume_pathname]

FIND_ORPHANS finds all uncataloged permanent objects in a local volume. It uses or creates a directory ORPHANS in the root of the volume and enters the names of all objects not cataloged elsewhere. Uncataloged directories are found first, so no redundancy occurs.

The user of this command must have read permission to all directories on the volume. If some directory is not readable, every object under that directory will be cataloged in the ORPHANS directory. In addition, the user must either have permission to create the ORPHANS directory or to catalog objects in ORPHANS when it already exists.

The objects cataloged by FIND_ORPHANS are given sequential names like F1, F2, etc., and can be moved using MVF to a directory of the user's choice.

This command is useful for finding objects that are lost by a broken directory. It should be run only on a quiescent node: i.e., one not connected to the network (use NETSVC -N to disable network communications) and not actively running any processes other than the one performing the FIND ORPHANS operation.

ARGUMENTS

volume_pathname

(optional)

Specify the name of the volume to be searched. The volume must be physically attached to your node; you may not find orphan objects on volumes elsewhere in the network.

Default if omitted: search node boot volume

OPTIONS

-V[ERIFY]

Verify only; don't catalog any orphans

EXAMPLES

```
$ find orphans
11EE936C.50000105
                   ->
                       f1
1216E28E.40000105 ->
                       f2
12A2BC34 . 40000105
                       f3
12B782DC.40000105 ->
                       f4
12B78321.50000105 ->
                       f5
12B78353.60000105
12B783EF.00000105
                       f7
12B784E3.90000105
                       f8
12B7863C.30000105
                       f9
12C18DBE.40000105
                  ->
                       f10
12F98201.40000105
13452895.80000105 ->
```

140090B4.40000105 -> f13 140090F4.E0000105 -> f14 15322D3A.70000105 -> f15 17872C66.50000105 -> f16 Number of orphans: 16 \$ 1d /orphans -a

Directory "/orphans":

sys	type	blocks	current			
type	uid	used	length	attr	rights	name
file	rec	2	1462	P	p-ndwrx	f1
file	nil	0	0	P	p-ndwrx	f10
file	obj	1	58590	P	p-ndwrx	f11
file	nil	4	4096	P	p-ndwrx	f12
file	nil	4	4096	P	p-ndwrx	f13
file	nil	4	4096	P	p-ndwrx	f14
file	uasc	0	245	P	p-ndwrx	f15
file	mbx	17	280172	P	p-ndwrx	f16
file	nil	0	0	P	p-ndwrx	f2
file	nil	0	0	P	p-ndwrx	f3
file	obj	66	65862	P	p-ndwrx	f4
file	obj	134	135724	P	p-ndwrx	f5
file	rec	9	8412	P	p-ndwrx	f6
file	obj	115	116188	P	p-ndwrx	f7
file	mbx	15	278636	P	p-ndwrx	f8
file	nil	0	0	P	p-ndwrx	f9

16 entries, 371 blocks used.

FLEN (FILE_LENGTH) -- Count lines, words, and characters in a file.

FORMAT

FLEN [options] [pathname ...]

FLEN prints the number of lines, words, and characters in each of the named files. A word is defined as any sequence of characters delimited by tabs, spaces, and NEWLINEs. If more than one file is specified, totals for all the files are printed, also.

ARGUMENTS

pathname

(required)

Specify input file. Multiple file names and wildcarding are

permitted.

Default if omitted: read standard input; suppress total counts

OPTIONS

If no options are specified, all counts are reported.

-L

Print only line counts.

-W

Print only word counts.

-C

Print only character counts.

Options may be mixed to achieve the desired reporting results.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ flen -L -C mary Print the number of lines and characters in the file 'mary'

FMC (FORMAT_MULTI_COLUMN) -- Format text into multiple columns.

FORMAT

FMC [options] [pathname ...]

FMC reads the named files and formats them into multiple columns on standard output. Each input line is placed in one column of an output line; input lines which are longer than the output column width are truncated. This command is useful to format text which is already in the form of a column or list.

ARGUMENTS

pathname

(optional)

Specify input file. Multiple pathnames are permitted, separated

by blanks.

Default if omitted: read standard input

OPTIONS

The options control output format. If no options are specified, the default output format is:

	number of columns	2	
	page length	55	
	column width	60	
	gutter width	8	
-C n	Specify n columns.	Default: 2.	
-L n		n in n lines. FMC produces eparators between the pages.	
-W n	Specify column wide characters are trunc	th in n characters. Input line ated. Default: 60.	es longer than n
-G n	Specify gutter wi between columns. I	dth in n spaces. The gut Default: 8.	ter is the space
-D n	set to n characters number of column	minal as output device. The and the page size is set to a and the gutter width a nt of information on the screen	24 lines. The re computed to

EXAMPLES

\$ crefs sample | fmc -c 3 -w 22 -g 4

This command line first produces a cross-referenced list of all the symbols in the file SAMPLE, then formats the report in a 3-column list.

FMT (FORMAT_TEXT) -- Format a text file.

FORMAT

FMT [options] [pathname ...]

FMT is a general purpose text formatting program, allowing you to arrange output text according to formatting directives embedded in the input file or typed on standard input.

By default, formatted text is written to standard output. You may redirect it to a file with the -OUT option.

ARGUMENTS

pathname		
(optional)	Specify input file to be formatted. wildcarding are permitted; however,	FMT will concatenate
	multiple files prior to formatting. If Fl specified input files, control shifts to sta	

Default if omitted: read standard input

OPTIONS

-F n	Begin output at the first page numbered n.
-T n	Terminate output at the first page numbered higher than n.
- S	Stop before printing each page, including the first. This option is useful for paper manipulation. The prompt "Type return to begin a page" is issued only once, before the first page.
-PO n	Page Offset. Shift the entire document n spaces to the right.
-LF	List names of files as they are processed.
-OUT pathname	
	Specify output file. If this option is omitted, formatted text is written to standard output.

EXAMPLES

\$ fmt	mary	-out	mary.formatted	-ро	9	Format "mary" with a page offset
\$						of 9 spaces, and write the
						results to "mary.formatted".

USING FMT

Input consists of text lines, which contain information to be formatted, intermixed with FMT request lines, which tell the program how to format the text lines. Request lines must begin with a special control character, normally a period. The Request Line Summary that follows this section lists the FMT commands.

Line Endings

If you use the fill (.fi) command, FMT adds as many words to an output line as will fit. FMT determines the line break; the <RETURN> you type when entering text does not establish line breaks on the output.

Justification

You can turn right justification on and off with the .ju and .nj commands. Strings of embedded spaces are retained so that the output line contains at least as many spaces between words as the input line. However, spaces at the beginning of input lines are output without modification.

Line Breaks

In certain cases, FMT generates breaks. A break consists of cancelling right justification for a particular line (even while you are using .ju) and beginning the next input line on a new output line.

Breaks are caused by an empty input line, by an input line that begins with a space, or by certain commands (see the "break" column on the Request Line Summary chart at the end of this section). You can also cause a break by using the .br command. Breaks are used most often to create paragraph breaks.

Tabbing

You may use FMT's tabbing feature as an aid in creating tables, lists, and other items which require multiple columns. The .ta command, followed by a list of integers, sets tabs stops at the given columns (i.e., .ta 5 10 sets tabs at columns 5 and 10.) The .tc command declares the tab character: for instance, .tc \ would cause FMT to move output to the next tab stop whenever a backslash (\) is encountered in the input text. Finally, the .rc command establishes a "replacement character" for use in tabbing operations. The replacement character is printed from the current point to the next tab stop. This can be useful when creating a table of contents, for instance.

The following input file:

```
.ta 50 Set a tab stop at column 50.
.tc \ Declare the tab character (\).
Section\Page
.rc Declare the replacement character (.)
.sp
Chapter 1\1
Chapter 2\15
Chapter 3\25
.rc Return the replacement character to blank.
```

would produce:

FMT (FORMAT_TEXT)

Section		Pag	zе
Chapter	1		
	2		
Chapter	3		

Running Titles

Running titles may appear at the top and bottom of every page. A title line is a single line comprising three textual fields: the first is placed flush with the left margin, the second is centered, and the third is placed flush with the right margin. The first nonblank character in the title is used as the delimiter to separate the three fields. FMT replaces any pound signs (#) in a title with the current page number, and replaces any percent (%) characters with the current date.

Number Registers

FMT provides 26 number registers named a through z. The .nr command defines a number register. The command

.nr x m

sets number register x to m;

.nr x +m

increments number register by m; and

.nr x -m

decrements x by m. To insert the value of number register x in the text or a command line, specify @nx. Number registers are useful for running counts. For example, table numbers in this manual are stored in number register t. The register is set to zero at the beginning of every chapter and is incremented each time a new table appears.

Fixed Spaces

You can insert a fixed number of spaces anywhere in the text except header and footer titles. Fixed spaces are most often used to override the justification command (.ju), which inserts extra spaces to justify a line.

By default, the pound sign character (#) represents a fixed space. For example, placing three pound signs in a string results in exactly three spaces. The phrase

Plot###data

in the source file would appear in the formatted output as:

Plot data

To print an actual pound sign, "escape" it with an at sign. For example:

16@#100 appears as 16#100

To change the default to another character, use the .sc command.

In-Line Boldfacing and Underlining

In-line flags allow you to mark text to be underlined or boldfaced, without giving the command on a separate line. The flags are:

```
{_underlines these words_}
{!boldfaces these words!}
```

These underlining and boldfacing flags can be used instead of the .ul, .cu, and .bd commands, which must be on lines separate from the text to which they apply.

The underlining flags apply to the entire string they surround; hence embedded spaces (including fixed spaces) will be underlined.

Each of the character sequences {_, {!, _}}, and !} is treated as a unit. Therefore you need only one at sign, preceding the sequence, to nullify its special meaning as a flag.

Defining New Commands

You can define your own commands with the .de command. For example, the following sequence defines a paragraph command named .pg.

.de pg

.sp

.ti +3

.en

Defined commands can be invoked with arguments, separated by blanks or tabs. Within a command definition, arguments are referenced using ^1, ^2, etc., up to a maximum of nine arguments. Omitted arguments default to the null string, and ^0 refers to the command name itself. For example, the following version of the paragraph command uses the argument to determine the amount of indentation.

.de pg

.sp

.ti +^1

.en

This command could be invoked by:

.pg 3

to get the same effect as the previous version.

Inserting Files

The .so command,

.so filename

causes the contents of the named file to be inserted in place of the command line. You can nest .so commands.

FMT (FORMAT_TEXT)

DIAGNOSTICS

"string": unrecognized command ignored.

Your input file contains an invalid FMT request.

invalid number register name

Names of number registers must be a through z.

missing name in command definition

No name was supplied with the .de command.

.so commands nested too deeply

The limit for nesting included source files has been exceeded.

too many characters pushed back

The buffer holding input characters has been exceeded.

REQUEST LINE SUMMARY

Request Initial Default Break Meaning

.#			no	Ignore this line. Precede comment lines with this symbol.
.bd n		n=1	no	Boldface the next n lines
.bp n	n=1	n=n+1	yes	Begin new page and number it n
.br			yes	Break
.cc c	c=.	c=.	no	Control character becomes c
.ce n		n=1	yes	Center the next n input lines
.cu n		n=1	no	Continuously underline next n input lines
.de xx			no	Command xx; ends at .EN
.ef /l/c/r			no	Foots on even pages are l(eft), c(enter), r(ight). '#' and '%' produce page number and date, respectively.
.eh /l/c/r			no	Heads on even pages are l(eft), c(enter), r(ight). '#' and '%' produce page number and date, respectively.
.en			no	Terminate command definition
.fi	yes		yes	Begin filling output lines
.fo /l/c/r		×	no	Foot titles are 1(eft), c(enter), r(ight) '#' and '%' produce page number and date, respectively.
he /1/c/r			no	Head title is l(eft), c(enter), r(ight)
, . , . , . , .				'#' and '%' produce page number and date, respectively.
in n	n=0	n=0	yes	Set left margin to column n+1
.ju	yes	yes	no	Begin justifying filled lines
.ls n	n=1	n=1	no	Set line spacing to n
.m1 n	n=3	n=3	no	Space between top of page and head
.m2 n	n=2	n=2	no	Space between head and text
.m3 n	n=2	n=2	no	Space between text and foot
.m4 n	n=3	n=3	no	Space between foot and bottom
ne n	п О	n=0	y/n	Need n lines; break if new page
.nf	no		yes	Stop filling
.nj	no		no	Stop justifying
.nr x m	m=O	m=0	no	Set number register x to m,
. 111 . 111	ш-0	m-0	110	-m, +m for decrement, increment
.of /1/c/r			no	Foots on odd pages are l(eft), c(enter),
.01 /1/0/1			110	r(ight). '#' and '%' produce page number and date, respectively.
.oh /l/c/r			no	Heads on odd pages are l(eft), c(enter), r(ight). '#' and '%' produce page number
m1 m	7-66	7-66	20	and date, respectively. Set page length to n lines
.pl n	n=66	n=66	no	Set page offset to n spaces
.po n	n=0	n=0	no	Tab replacement character is c
.rc c		CE	no	
.rm n	n=65	n=65	no	Set right margin to column n Change fixed space character to c
.sc c	c=#	C=#	no	•
.so file		n-1	no	Switch input to file Space n lines, except at top of page
.sp n		n=1	yes	
.st n		n=0	yes	Space to line n from top; -n spaces to line n from bottom
ta ni n2			no	Set tab stops at columns n1, n2,
.tc c		-	no	Tab character is c
.ti n		n=0	yes	Temporarily indent next output line n spaces
.ul n		n=1	no	Underline words in the next n input lines

FMT (FORMAT_TEXT)

IN-LINE FLAGS

{_c_}	Underline characters enclosed in braces
{!c!} no	Boldface characters enclosed in braces
@nc no	Replace with value in number register c
# no	Insert literal blank

KEY

- n denotes numerical values
- t denotes titles
- c denotes single characters

Signed numbers signify relative changes to a quantity; unsigned numbers signify absolute settings. Unless otherwise noted, omitted n fields set the value to 1, omitted t fields are empty, and omitted c fields restore the default character.

FOR -- Execute a FOR statement.

FORMAT

FOR var_name := int_expr [TO int_expr] [BY int_expr] command... ENDFOR

FOR var name IN string expr [BY {CHAR|WORD|LINE}] command... ENDFOR

FOR allows you to build a control structure that executes commands repeatedly as long as the result of a Boolean test is TRUE. The FOR command has two formats: one for assigning and testing integer expressions, and one for assigning and testing string expressions.

In the integer form, the (optional) TO and BY clauses permit you to specify ranges and increment values, respectively. For example, you might want to loop 5 times by saying

FOR a := 0 TO 10 BY 2

If you do not specify "BY int_expr", the default increment is 1. If you do not specify "TO int_expr", you will probably want to increment the variable manually inside the body of the loop. You should also put a test condition inside the loop (and probably use an EXIT to get out) or else you risk looping forever.

In the string form, the (optional) BY clause allows you to control the string assignment operation. If you specify "BY WORD" (the default), each word (a sequence of non-blank characters) in 'string_expr' is assigned to 'var_name' one at a time until 'string_expr' is exhausted. You may also assign string values a character at a time, or a line at a time, by using the "BY CHAR" and "BY LINE" clauses, respectively.

ARGUMENTS

var	name

(required) Specify the name of the Shell variable whose value is to be

assigned and tested.

int expr

(required) Specify any valid expression that returns an integer value.

string_expr

(required) Specify any valid expression that returns a string value.

command...

(required) Specify the command to be executed as long as the FOR test

returns TRUE. This may be a Shell command, a Shell script, a variable assignment, or any other valid Shell operation. Multiple commands are permitted; separate them with semicolons or

NEWLINE characters.

EXAMPLES

1. The following example demonstrates the advantages of a FOR loop over a WHILE loop in one instance. Assume these line appear in a Shell script.

```
#
# A loop using WHILE.
#
eon
a := 0
while ((^a <= 10)) do
    args ^a
    a := ^a + 2
enddo
#
# The same loop using FOR.
#
FOR a := 0 TO 10 BY 2
    args ^a
ENDFOR
#
# End of script.</pre>
```

2. This example assigns three names to a variable.

```
# # Script FILE_NAME
#
eon
FOR file IN "foo bar zap" BY word
    args ^file
ENDFOR
#
# End of script.
```

Execution produces:

```
$ file_name
foo
bar
zap
$
```

FPAT (FIND_PATTERN) -- Find a text pattern in an ASCII file.

FORMAT

FPAT [options] [pathname... -P] reg_expr ...

FPAT searches its input file(s) for lines matching the specified regular expressions and writes them to standard output or the file specified.

ARGUMENTS

reg_expr...

(required)

One or more regular expression patterns. By default, a line that contains any of these expressions matches and is written to standard output. For a description of regular expressions used for pattern matching, see the chapter on DM basics. Patterns containing embedded spaces or Shell special characters must be enclosed in quotation marks.

pathname -P

(optional)

Specify name of file to be searched. If you specify a pathname with this argument, you must follow it with "-P" to separate the pathname(s) from the search patterns on the command line. Multiple pathnames and wildcarding are permitted.

Default if omitted: read standard input

OPTIONS

If no options are specified, any line that matches any one of the regular expressions is considered a matching line.

-OUT pathname

Write output to specified file. If input file names were specified, output filename can be derived. If this option is not specified, matching lines are written to standard output.

-A Select only lines that match ALL regular expressions, in any order.

-X Select only lines containing NONE of the regular expressions.

-C Write only a count of matching lines, not the lines themselves.

-I Ignore cases for search (i.e., become case-insensitive).

-L Write line number with each line that matches the regular expression.

-M n Set the maximum number of search lines to n (a decimal value).

FPAT terminates after searching n lines.

FPAT (FIND_PATTERN)

-LF Display the name of the file being examined before searching its lines.

-LM Similar to -LF, but display the name(s) of only those file(s) which contain matches for the regular expression.

-RM n Set maximum number of matches to be reported for this execution of FPAT.

-RMF n Set maximum number of matches to be reported for each file being searched.

Summary of Regular Expression Notation

```
Literal character
         Any character (except newline)
%
         Beginning of line
$
         End of line
        Character class (any one of these characters)
[...]
[~...]
        Negated character class (all characters except those in brackets)
[c1-c2] Any single character in the range c1 through c2
        Escaped character (e.g., e%, e[, e*)
@n
         Newline
et
         Tab character
         Closure (zero or more occurrences of previous pattern)
        Tagged pattern
```

EXAMPLES

1. Assume the file "text" contains:

now
is
the
time
for
all
good

Then the command,

```
$ fpat -x -m 5 -l text -p o produces ...
( 2) is
( 3) the
( 4) time
$
```

2. \$ fpat text?* -p the

Search for the string "the" in all files whose names begin with "text".

3. \$ fpat text?* -p the -out =.out

Search for the string "the" in all files whose names begin with "text", (i.e., "text", "text1", "text_file", etc.) and write the output to the files "text.out", "text1.out", "text_file.out", etc.

FPATB (FIND PATTERN BLOCK) -- Find blocks of text containing patterns.

FORMAT

FPATB [options] [pathname... -P] reg_expr ... [-OUT pathname]

FPATB reads blocks of text from its input files and writes them to its output file(s) so they meet the specified matching criteria. By default, blocks of lines are separated by an empty line or by a line containing only blanks. FPATB is similar to FPAT (FIND_PATTERN) except that if a pattern is found, the entire block of lines is copied to output, rather than only the line in which the pattern occurs. Thus, it is useful for searching mailing lists, bibliographies, and similar files, where several lines are grouped together to form cohesive units.

ARGUMENTS

reg_	expr
(regi	ired)

Specify regular expression to be used for matching search. Each expression defines a text pattern, and you can specify up to nine expressions with each FPATB command. FPATB is case-sensitive; for example, "a" is different from "A". For a description of regular expressions used for pattern matching, see the chapter on DM basics.

pathname -P (optional)

Specify name of file to be searched. If you specify a pathname with this argument, you must follow it with "-P" to separate the pathname(s) from the search patterns on the command line. Multiple pathnames and wildcarding are permitted.

Default if omitted: read standard input

OPTIONS

If no options are specified, any block containing a line that matches any one of the regular expressions is considered a matching block.

-OUT pathname

Write output to specified file. If input file names were specified, output filename can be derived. If this option is specified, it must be the last option on the command line (i.e., it must follow any regular expressions specified). If this option is not specified, matching lines are written to standard output.

- -A Select only blocks containing lines that match ALL regular expressions, in any order.
- -X Select only blocks containing NONE of the regular expressions.
- -C Write only a count of matching lines, not the lines themselves.

-B reg_expr1

Specify 'reg_expr1' as the block separator, instead of a blank or empty line. Text blocks begin at lines containing 'reg_expr1'. If -B is specified and -E is not, 'reg_expr1' begins and ends the block.

-E reg expr2

Specify 'reg_expr1' to start a block and 'reg_expr2' to end a block. Note that the -E option is used only in conjunction with the -B option.

-L n

Write only the first n lines of selected blocks. If a block contains fewer than n lines, this option pads the output block with blank lines.

-LF

Display the name of the file being examined before searching its

Summary of Regular Expression Notation

c Literal character

? Any character (except newline)

% Beginning of line

\$ End of line

[...] Character class (any one of these characters)

[~...] Negated character class (all characters except those in brackets)

[c1-c2] Any single character in the range c1 through c2

ec Escaped character (e.g., e%, e[, e*)

en

Newline

et Tab character

Closure (zero or more occurrences of previous pattern)

{...} Tagged pattern

EXAMPLES

\$ fpatb address_list -p 01824 -out zip_list Locate text blocks with
the string "01824" in
the file "address_list"
and write the results
to "zip list".

FPPMASK (FLOATING_POINT_MASK) -- Set/display floating-point error mask.

FORMAT

FPPMASK [options]

FPPMASK sets or displays the state of the floating-point package error mask for a process. The error mask specifies some of the conditions that constitute a floating-point exception for the process.

OPTIONS

If no options are specified, the current floating-point error mask condition is displayed.

-D condition ...

Disable 'condition' (see below). Both conditions may be specified.

-E condition ...

Enable 'condition' (see below). Both conditions may be specified.

'condition' may be either of the following:

LOS

Loss of significance. This condition occurs when subtracting floating-point values that are exactly equal.

UNDR.

Underflow. This condition occurs when a floating-point result is too small to be represented. For single-precision values, this error occurs at about 0.118E-37. For double-precision values, the error occurs at about 0.223E-307.

Both conditions are initially disabled when a process is created.

EXAMPLES

1. \$ fppmask Display current settings. LOS (loss of significance): disabled

UNDR (underflow): disabled

2. \$ fppmask -e los undr Enable both LOS and UNDR conditions.

3. \$ fppmask -d undr Disable UNDR condition.

FSERR (FIND_SPELLING_ERRORS) -- Find spelling errors.

FORMAT

FSERR [pathname ...] [options]

FSERR copies the named files line-by-line to standard output, while looking up each word in a dictionary. If it finds any spelling errors on a line, or if it finds words that are not in the spelling dictionary, FSERR prints the line containing the questionable word and asks whether or not the word is spelled correctly. If you indicate that the word is misspelled, you are prompted for the correct spelling. FSERR corrects the spelling on standard output and continues.

Specify file containing text to be checked. Multiple pathnames

ARGUMENTS

pathname (optional)

		Default if omitted: read standard input
OPTI	ONS	
	-F	Process words just after a period ('.') in column 1 (i.e., FMT directives). The default is to ignore such "words".
	-N	Process digits. The default is to ignore digits.
	-U	Underline misspelled words instead of prompting for correction or verification.
	-S	Collect and print statistics on dictionary use.
	-C pathname	Write words that are not in the dictionary, but are correctly spelled, into 'pathname'.
	-D pathname	Add the words in the file 'pathname' to the dictionary used for this run.

are permitted separated by blanks.

FILES

/SYS/DICT	The dictionary file is an alphabetical list of words that we supply. New words
	can be added to the dictionary with an editor, and ordered with SRF
	(SORT_FILE). Note that the word list must be in alphabetical order.

/SYS/DICTDX This file, an index to the dictionary file, is created the first time FSERR is used. If you change the dictionary, you must also delete the index, so that it can be recreated to reflect the changes. (FSERR automatically creates a new

FSERR (FIND_SPELLING_ERRORS)

index if the file /SYS/DICTDX is missing.) If you do not delete the old index after you modify the dictionary, FSERR gives incorrect results. Creating /SYS/DICTDX takes a long time.

FST (FAULT_STATUS) -- Print fault status information.

FORMAT

FST [options]

FST prints information about the most recent fault that occurred in the process. The information always includes the fault status, the program counter (PC) at which the fault occurred, and a textual description of the error (as reported by the system call ERROR_\$PRINT).

This command is intended for system-level debugging.

If you are using a Peripheral Bus Unit (PBU) device, you can get fault information by using the option "-U" (see below).

OPTIONS

-R	Print the content	s of the CP	J general	registers	when	the fault
	occurred.					

-S	Print t	the sup	erv	isor	PC,	entry	contr	ol	block (ECB), and	status
	register	(SR)	if	the	fault	occi	ırred	in	supervisor	mode.	This
	option i	s igno	red	if th	e faul	lt occi	urred	in	user mode.		

-A	Print	all	available	fault	information.	(Prints	the	same
	informa	tion	as both -S	and -F	2.)			

Print the same information as both -S and -R for faults caused by the PBU interrupt handler for unit n.

EXAMPLES

\$ fst -a

-Un

Fault Diagnostic Information
Fault Status = 00120010:

process quit (from OS / fault handler)

User Fault PC = 000157FC

D0-D7: 00120010 00000000 00000002 FFFFFFFE 00000008 00000006 00000182 00000004 A0-A7: 0020A452 00E2F22E 0020A3D4 0020A450 00E2F174 0000C92C 002746B4 002746AC

Supervisor ECB = 00000000 Supervisor SR = 0000 Supervisor PC = 00000000

HELP -- Provide help on Shell and DM commands.

FORMAT

HELP [topic [subtopic]]

This feature provides information on Shell and DM commands and miscellaneous system services by opening a window to display the file that you request. For a list of subjects in the HELP library, type:

\$ HELP INDEX

Access to this facility is also provided through the <HELP> key on DN300 keyboards.

ARGUMENTS

topic

(optional)

Specify the name of the command or topic for which you desire

help.

Default if omitted: display introductory information

subtopic

(optional)

Specify subtopic to be viewed. For example,

\$ help shell commands

displays a topical index of Shell commands, while

\$ help shell

displays general information about the Shell.

Default if omitted: no subtopic displayed

HLPVER (HELP VERSION) -- Provide HELP support for Shell scripts.

FORMAT

HLPVER script name version ^1

HLPVER provides access to the DOMAIN HELP system utilities that support the standard command options -HELP, -VERSION, and -USAGE for Shell commands. By placing the HLPVER command inside a Shell script, you can allow users of the script to specify these three standard command options and receive meaningful output.

HLPVER looks for help information in a file called /SYS/HELP/script_name.HLP. HELP files have special information at the top that HLPVER uses. This information must follow a standard format. The following example shows the header of the HELP file for the WD (WORKING DIRECTORY) command.

1.1; wd (working_directory), revision 1.1, 81/06/27 WD (WORKING_DIRECTORY) -- Set or display the current working directory. usage: WD [pathname]

All HLPVER output goes to standard output (normally directed to the process transcript pad). HLPVER returns the first line of the HELP file in response to -VERSION. It returns the second line through the first blank line in the file in response to -USAGE. It returns the entire file in response to -HELP.

Any user file placed in the /SYS/HELP directory is also available to the HELP command for display in a standard HELP window. Thus the file /SYS/HELP/MARY.HLP can be viewed with \$ HELP MARY regardless of whether or not you are using HLPVER inside the MARY script. HLPVER is solely for the purpose of enabling the three standard command options mentioned above.

ARGUMENTS

script name

(required)

Specify the name of the script for which HELP is to be provided. The name is the right-most leaf in the pathname, not the entire pathname of the script. HLPVER uses this name to construct the pathname for the HELP file to be returned (i.e., /SYS/HELP/script_name.HLP).

version

(required)

Specify the version number of the Shell script. HLPVER compares this number to the version number in the HELP file (the first characters in the file up to the first semicolon) and returns an error if they do not match. This allows you to coordinate versions of the script and the HELP file.

^1

(required)

Pass in the desired option from the command line. "^1" must appear literally so that HLPVER can tell what information to return (-HELP, -VERSION, or -USAGE). See the example below.

EXAMPLES

Assume that the following lines appear in a file called "test_script":

#
Example script showing HLPVER usage.
#
hlpver test_script 1.0 ^1
args "Please enter ..."

End of script

When the user types:

\$ test_script -help

HLPVER returns the contents of /SYS/HELP/TEST_SCRIPT.HLP to the transcript pad. Likewise, when the user types:

\$ test_script -version

HLPVER returns the first line of the HELP file containing the version number.

HPC (HISTOGRAM_PROGRAM_COUNTER) -- Program counter histogram.

FORMAT

HPC [options] pathname [args...]

HPC produces a histogram of the program counter during program execution, thus helping you locate the most time-consuming portions of your program.

While your program is executing, HPC samples the program counter at regular intervals, gathering a set of data points. Each data point is the address at which the program was executing -- that is, the value of the program counter -- when the sample was taken.

When execution of your program has ended, HPC displays statistics and a histogram (bar graph) of the program counter. Each bar corresponds to an area of program memory. The length of the bar indicates how much time the program spent executing in the corresponding area.

While HPC and your program are executing, serial line 3 (SIO3) (line 2 on DN3xx systems) is not available for output.

ARGUMENTS

pathname	
----------	--

(required) Specify the name of the program to be evaluated.

args

(optional) S

Specify any arguments to be passed to the program "pathname". These are not processed by HPC, but passed directly to your program.

Default if omitted: no arguments passed

OPTIONS

If no options are specified, a histogram is produced for the entire program, with 500 samples taken per second.

-LOW x Specify lowest address ('x') to be included in the histogram. 'x'

must be a hexadecimal value. If this option is omitted, the

histogram starts at the beginning of the program.

-HIGH x Specify highest address ('x') to be included in the histogram. 'x' must be a hexadecimal value. If this option is omitted, the

must be a hexadecimal value. If this option is omitted, the histogram continues to the end of the program.

By limiting the range of addresses in the histogram with -LOW

and -HIGH, you can study a specific part of your program, such as an I/O routine.

-RATE n Specify how many times ('n') HPC samples the program counter

per second. 'n' must be a decimal number in the range 5 to

2000. The default is 500 samples per second. A higher rate results in a more accurate histogram, but tends to slow program execution.

EXAMPLES

In the following example, HPC samples the LD (LIST_DIRECTORY) program. Note the use of the pathname /COM/LD to specify the LD program in the /COM directory. The second argument, /LATEST/COM, is passed on to LD as an argument.

The output at (1) is a section map, showing the addresses and sizes of the program sections in LD.

At (2) is the output of the LD program itself -- in this case, a listing of the directory named by /LATEST/COM. (Note that /LATEST is a link that resolves to //US/LATEST.)

The two lines at (3) summarize HPC parameters and statistics, as follows:

PID The ID of the process that HPC is monitoring.

Low and High The range of addresses in the sample. These addresses are derived from those specified with the -LOW and -HIGH options, after

rounding.

Bucket size

The number of addresses represented by each bar in the histogram.

HPC divides the range of addresses in the sample into 512 "buckets."

Each "bucket" is, in effect, a receptacle for data points in one

five-hundred-twelfth of the address range in the sample.

Interrupts The number of times HPC sampled the program counter.

Hits The number of data points with the correct PID that were also within

the Low and High limits.

Miss low The number of data points with the correct PID, but below the Low

address.

Miss high The number of data points with the correct PID, but above the High

address.

Miss PID The number of data points with another process's PID.

Finally, the histogram appears at (4). The output has been shortened for this example, as indicated by the ellipsis points. The histogram lists the routines in the program, in order of their starting addresses. HPC displays the name and starting address of each routine, followed by the histogram bars that represent addresses within the routine. Each bar is prefaced by the line numbers it represents and by a percentage. The line numbers correspond to one bucket, and the percentage indicates the percentage of hits that fell into the bucket as compared to the total number of interrupts.

Note that if many routines fall within the range of a single bucket (either because of small routines or large buckets), all their names are listed before the bar. In this case, details on the activities within the individual routines are not available. For more

detailed information, use the -LOW and -HIGH arguments to "zoom in" on individual routines.

Following the percentage is the bar itself. The longer the bar, the more time the program spent executing the statements in the group.

Finally, at (5), HPC displays a line labeled "Totl Pcnt." This number is the ratio of hits to interrupts, expressed as a percentage. A low percentage here (such as the 8% in the example) indicates that the program uses relatively little CPU time, and that you have little to gain by trying to further optimize.

\$ hpc /com/ld /latest/com
Section Map:

- # Location Size Name
- 1 000E8040 00001D5C PROCEDURE\$
- 2 000C680A 000002EA DATA\$
- 3 000E9D9C 00000454 DEBUG\$

Directory "//us/latest/com":

aqdev	arcf	args	asm	bind	bldt	catf	chn
clstr	cmf	cmsrf	cmt	cpboot	cpf	cpt	crd
crefpas	crefs	crf	crl	ctnode	ctob	date	db
dlf	dll	dlt	dmtvol	ed	edstr	emt	esa
flen	fmc	fmt	fpat	fpatb	fppmask	fserr	fst
help	hpc	invol	lamf	las	lcnode	ld	lkob
lopstr	lvolfs	macro	mtvol	mvf	nd	netstat	netsvc
os	pagf	pas	prf	protd	protf	prsvr	rbak
revl	rfc	rldev	rwmt	rwvol	sald	salvol	sh
siotf	srf	stcode	tb	tctl	tee	tlc	tpm
uctnode	uctob	ulkob	wbak	wd			

Directory contains 95 entries. PC Histogram

3 PID= 9; Low= 000E8040; High= 000EA03F; Bucket size= 00000020
Interrupts= 2462; Hits= 193; Miss low= 67; Miss high= 477; Miss PID= 1725

```
SORT NAMES (000E8B9A)
0297-0298: 0% X
0298-0298: 1% XXXXXXXXXXXXXXXX
0300-0301: 1% XXXXXXXXXXXXXXXXXX
0302-0302: 0% XXXXXXXXXX
0303-0303: 0% XXXXXXXXX
SHOW LINK TEXT (000E8C8A)
   -0321:
         0% X
          0%
0322-0322:
0323-0325:
          0%
 000E8CE0:
          0%
 000E8D00:
          0%
LDIR $WS (000E8D24)
   -0337:
         0%
          0%
0339-0339:
          0%
0341-0341:
0343-0346:
```

${\tt HPC}\;({\tt HISTOGRAM_PROGRAM_COUNTER})$

IF -- Execute a conditional statement.

FORMAT

IF condition THEN command 1 ... [ELSE command 2 ...] ENDIF

IF executes a conditional statement depending on the results of a Boolean test. You can extend the IF command over several lines if you use it interactively or in a Shell script. When you use IF interactively, and extend the command over more than one line, the Shell prompts you for each new line of the command with the \$_ prompt (refer to the example below).

ARGUMENTS

condition

(required)

Specify a command or program to execute and test for truth, or specify a variable expression or Boolean variable to test for truth. "Truth" usually means that the command executes successfully (without any errors), or that the Shell variable expression or Boolean is "true". (Specifically, this argument is evaluated TRUE if it returns an abort severity level of 0 (zero).)

Refer to the DOMAIN System User's Guide for more information on Shell variables.

command_1

(required)

Specify command or program to execute if 'condition' returns TRUE.

command_2

(optional)

Specify command or program to execute if 'condition' returns FALSE (i.e., a severity level greater than zero).

EXAMPLES

```
1 $ IF eqs a a
$_ THEN args "a is a"
$_ ELSE args "Aristotle was wrong."
$_ ENDIF
a is a
$
```

```
2. IF eqs ^2 '-c'
THEN pas ^1
        bind ^1 bin library -b ^1
ELSE bind ^1 bin library -b ^1
ENDIF
```

Example 2 might appear in a Shell script. These lines will compile the Pascal module named by the first argument (^1) if the second argument (^2) is '-c'. Then it will bind the module with 'library'. If the second argument is not '-c', or if there is no second argument, the command simply binds the module.

INLIB (INSTALL_LIBRARY) -- Install a user-supplied library.

FORMAT

INLIB pathname...

INLIB installs a library at the current Shell program level; it remains installed until the Shell that installed it exits. (To load a library that is used by all processes, see note below.) The newly installed library will be used to resolve external references of programs (and libraries) loaded after its installation. (Thus, previously loaded libraries and programs will NOT be affected.)

INLIB is an internal shell command.

NOTE: At Version 4.1 and later you can create a library that will be installed automatically in every process. This library resides in the file /LIB/USERLIB.PRIVATE. The procedure text in this library will be shared among all processes.

This library must be present at node startup time in order to be installed. After copying your library to /LIB/USERLIB.PRIVATE with the Shell command CPF (COPY_FILE), you must shut down the node and start it up again in order to use the library. Changes to the library also require rebooting the node to load the new routines.

Global names in /LIB/USERLIB.PRIVATE must not duplicate names used in DOMAIN libraries.

ARGUMENTS

pathname

(required)

Specify name of library file(s) to be installed. Multiple pathnames and wildcarding are permitted.

EXAMPLES

1. \$ inlib my lib

Install the library "my lib".

2. \$ inlib ?*.lib

Install all files in the current working directory with a ".lib" suffix.

INVOL (INITIALZE_VOLUME) -- Initialize disk volumes.

FORMAT

From AEGIS command Shell: INVOL

From Mnemonic Debugger: EX INVOL

INVOL initializes physical disk volumes, creates logical volumes, and maintains badspot lists. Once initialized, a volume can be mounted with the MTVOL (MOUNT_VOLUME) command, or can be used to bootstrap the operating system, providing it contains the necessary files.

Refer to the appendices for a description of INVOL's operation.

INVOL prompts for all required information.

LAMF (LAMINATE_FILE) -- Laminate files.

FORMAT

LAMF [pathname...] [-S string]

LAMF laminates the named files to standard output. That is, it concatenates the first lines of all input files, sequentially, and writes the result to standard output; and so on for the next input lines. If the files contain different numbers of lines, null lines are used for the missing lines in the shorter files.

NOTE: To insert a NEWLINE character between lines, use the escape sequence, @n, as a string argument. (See Example 4, below.)

ARGUMENTS

pathname

(optional)

Specify name(s) of file(s) to be laminated to standard output. Multiple pathnames are permitted, separated by blanks.

Default if omitted: read standard input for input lines. Use a hyphen (-) to specify standard input in a list of file names.

OPTIONS

-S string

Specify a string of text to be placed in each output line at the point it appears in the command argument list. 'String' may not exceed 300 characters. Strings containing embedded spaces must be in quotes ("").

EXAMPLES

1. \$ lamf mary fred

Laminate files "mary" and "fred" and write results to standard output.

2. \$ lamf jan - george

Laminate lines from "jan", standard input, and "george", in that order.

3. \$ lamf -S "A line from A: " a -S ", and from B: " b

would produce:

A line from A: first line from a, and from B: first line from b

Note that the text strings inserted are not bound in any way to the position of the pathname arguments: you may place strings wherever you please. Those strings that contain literal blanks must be enclosed in quotes, as above.

Escape sequences are valid in string arguments. For example:

4. \$ lamf mary -S @n fred

Insert a NEWLINE character between each line from mary and fred, thus interleaving the lines from the two files.

LAS (LIST_ADDRESS_SPACE) -- List objects mapped into the address space.

FORMAT

LAS [options]

LAS produces a list of objects mapped into the address space. Information printed includes the virtual address range, the starting address within the object, and its pathname if available (in that order).

This command is most useful for system-level debugging.

OPTIONS

If no options are specified, LAS lists the address space of the current process.

-ALL

List all address space, including that occupied by AEGIS.

-F[ROM] address

Begin listing at the hexadecimal address specified.

-T[O] address

End listing at the hexadecimal address specified.

-P[ROCESS] name

List addresses for the process named. Use the PST (PROCESS_STATUS) command to display the names of existing processes.

EXAMPLES

1. \$ las

		_				
	V	A Rai	nge	Obj	Start	Pathname
200	00		4 (7)			
800		_	17FFF		0	/sys/node_data/global_data
1800	00	_	2FFFF	•	0	/lib/pmlib
3000	00	-	37FFF	•	0	/lib/syslib.peb
3800	00	-	4FFFF	•	0	/lib/kslib
5000	00	-	57FFF	•	0	/lib/trait_type_lib
5800	00	-	67FFF	'	10000	/sys/node_data/global_data
6800	00	-	9FFFF	'	0	/lib/streams
A000	00	-	A7FFF		0	/lib/vfmt_streams
A800	00	-	BFFFF		0	/lib/error
C000	00	-	E7FFF		0	/lib/swtlib
E800	00	-	F7FFF		0	/lib/ftnlib
F800	00	-	FFFFF		0	/lib/pbulib
10000	00	-	127FFF		0	/lib/gprlib
12800	00	***	14FFFF		0	/lib/clib
15000	00	-	157FFF		0	/lib/lisp_initlib
15800	00	-	15FFFF		0	/sys/node_data/global_rws
16000	00	-	16FFFF		20000	/sys/node data/global data
17000	00	-	187FFF		0	/lib/shlib
18800	00	-	19FFFF		0	/lib/tfp

```
/lib/dialoglib
                          0
1A0000 -
           1BFFFF
                               /sys/node_data/ipc_data
/sys/node_data/global_data
1C0000 -
           1C7FFF
                          0
1D0000 -
           1D7FFF
                      30000
200000 -
           2AFFFF
                          0
                               -- temporary file --
                               /sys/node_data/dm_mbx
2B0000 -
           2B7FFF
                          0
                          0
                               /com/sh
2B8000 -
           2BFFFF
                               -- temporary file --
                          0
2C0000 -
           2C7FFF
                          0
                               /com/las
208000 -
           2CFFFF
2D0000 -
           2F7FFF
                      B0000
                               -- temporary file --
                          0
                               /help_area/worksite
BC0000 -
           BCFFFF
BD0000 -
           BDFFFF
                           0
                               /jtj
```

2944 KB mapped.

2. \$ las -from 188000

VA	Rai	nge Ob	j Start	Pathname
188000	_	19FFFF	0	/lib/tfp
1A0000	_	1BFFFF	0	/lib/dialoglib
100000	_	1C7FFF	0	/sys/node data/ipc data
1D0000	_	1D7FFF	30000	/sys/node_data/global_data
200000	-	2AFFFF	0	temporary file
2B0000		2B7FFF	0	/sys/node_data/dm_mbx
2B8000	-	2BFFFF	0	/com/sh
2C0000	-	2C7FFF	0	temporary file
2C8000	_	2CFFFF	0	/com/las
2D0000	_	2F7FFF	B0000	temporary file
BC0000	-	BCFFFF	0	/help_area/worksite
BD0000	-	BDFFFF	0	/jtj

1408 KB mapped.

3. \$ las -f 188000 -t 200000

VA Ra	inge Ob	j Start	Pathname
188000 -	19FFFF	0	/lib/tfp
1A0000 -	1BFFFF	0	/lib/dialoglib
1C0000 -	1C7FFF	0	/sys/node_data/ipc_data
1D0000 -	1D7FFF	30000	/sys/node_data/global_data

288 KB mapped.

LBR (LIBRARIAN) -- Combine object modules into a library.

FORMAT

```
LBR {-C | -UPD} library_pathname [module_pathname] [options] [-]
```

The librarian manages libraries of object modules. It adds, removes, or replaces modules in the library. As input, you must provide the pathname of a library you want to create or update, followed by an optional list of object module pathnames and processing options. As output, the librarian produces a new or updated library file.

You can use LBR in two ways: by entering complete LBR command strings, or by using the "-" (hyphen) option to ask the librarian to prompt you for multiple strings of module _pathname arguments and options. By using prompting you can perform several operations on object modules in the same library file, without entering LBR each time.

For a complete description of the librarian, see the DOMAIN Binder and Librarian Reference.

Prompting

The optional hyphen at the end of the command line requests the librarian to begin prompting. The hyphen is valid only on the line containing the LBR command, and must be the last item on the line. Note that prompting is also requested if the command line contains only the LBR command.

If you request prompting, the librarian processes the arguments and options on the current command line, then displays an asterisk (*) on standard output. In response to the asterisk, you can enter additional module pathname arguments and options. For example:

```
$LBR -C mylib.lib - <RET>
*file1.bin -DEL my_module <RET>
*file2.bin -L -REPL file3.bin
*<RETURN>
```

Prompting ends when you enter the -END switch or press <RETURN> in response to the asterisk. After prompting ends, the librarian finishes creating or updating the library file.

Comment Statements

You can include comments to an LBR command during a prompting session or in a Shell script. Comments must be delimited by braces, as in this example:

```
$LBR -UPD plot.lib
*plot_line.bin { Add PLOT_LINE procedure to library }
*{ Generate library directory }
*-L
*-END
```

The librarian ignores any comments when it processes the command line.

Librarian Errors

If a problem occurs during LBR execution, the librarian displays a message on error output. The message indicates the nature and severity of the problem. Error-level messages are issued for fatal conditions, which prevent the librarian from creating or updating a library file. Warning-level messages are issued for conditions that do not prevent the librarian from producing a library file, but the file's contents may not be what you were expecting.

ARGUMENTS

-C | -UPD library_pathname

(required)

The pathname of the library output file must be specified on the command line before you can specify any option that performs an operation on a library (such as adding to, extracting from, or reporting about a library). The -C (CREATE) or -UPD (UPDATE) option must be specified with the library pathname argument to indicate whether you want to create or update a library. Remember that only one library output file can be specified per execution of the librarian.

module_pathname (optional)

Specify an object module to be added into the library. Multiple pathnames and wildcarding are permitted. If omitted, no new object modules are added to the library.

OPTIONS

The following options instruct the librarian to perform various tasks. Note that some options apply directly to a library, while other options act on modules within the library. Default options are indicated by "(D)".

-DEL module

Remove an object module from the library. If a module of the given name cannot be found in the library, a warning is issued.

-EX module [-O pathname]

Extract the named module from the library. If the pathname modifier is specified with -O, the module will be copied to that pathname. Otherwise, the module is copied to a file having the same name as the module.

List a directory of the library contents to standard output. -L This report shows the name of each module in the library, with a list of section information and global declarations and references for each module.

Cause LBR to issue purely informational messages such as a -MSGS (D) summary of the number of errors and warnings that occurred.

Cause LBR to supress issuing purely informational messages. -NMSGS

Do not list global variables which are defined in the system (D) -NSYS library when generating a listing of global definitions and references with the -L option.

-REPL pathname

Replace, in the library, any modules found in the file specified by pathname. This option has an effect equivalent to first deleting all the modules found in pathname from the library, and then adding all the modules in pathname back into the library. The advantage gained by using -REPL is that you do not need to know the names of the modules in 'pathname'. Also, if you attempt to add a module to a library without using the -REPL option, and a module of that name does already exist, an error message is issued. If a module found in pathname does not already exist in the library, a warning message is issued.

-SYS

List global variables which are defined in the system library when generating a listing of global definitions and references with the -L option.

- (hyphen alone)

Request librarian prompting for further arguments.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

Refer to the DOMAIN Binder and Librarian Reference for detailed examples of LBR.

LCNODE (LIST_CONNECTED_NODES) -- List nodes connected to the network.

FORMAT

LCNODE [options]

LCNODE lists the nodes currently connected to the network. The list contains the ID of every node connected, the time at which the node was started, the current time, and the name of each node's entry directory.

This command reports only the nodes that respond within a preset time limit. Should a node be connected, but temporarily unable to respond within the specified time, it will not appear in the produced list.

OPTIONS

-M[E]	Request	information	about	your	node	only.	This	option
	displays tl	ne node id.						

-B[RIEF]	Request br	ief output.	LCNODE lists	only the entry dir	rectory
	name for ea	ch connected	d node. Note t	hat the entry direct	tory of
	a diskless no	de is the en	try directory of	its paging partner.	

-ID	When used with -BRIEF, display the node ID in addition	1 to the
	entry directory.	
	· ·	

-C[OUNT]	Request node count only. LCNODE lists only the number of
	nodes responding to its query.

-MAX[NODES] n Set a limit on the number of nodes you want to see, even if more

could have responded.

EXAMPLES

1. \$ lcnode

The node ID of this node is 21. 2 other nodes responded.

	ID	Boot	time	Current	time	Entry Directory	
:	17 27	1984/06/09 1984/06/09 1984/06/09 1984/06/09	13:52:02 12:53:28	1984/06/09 1984/06/09	16:06:13 16:06:07	//QUARTER	

2. \$ lcnode -me

The node ID of this node is 21.

LCNODE (LIST_CONNECTED_NODES)

3. \$ lcnode -b

//DOLLAR //QUARTER //NICKEL //QUARTER

- \$ lcnode -c
 466 other nodes responded.
- 5. \$ lcnode -c -b 466
- \$ lcnode -c -m
 The node ID of this node is 116A.
 466 other nodes responded.
- 7. \$ lcnode -b -id 21 //DOLLAR 17 //QUARTER 27 //NICKEL
 - 11 //QUARTER

LD (LIST_DIRECTORY) -- List contents of a directory.

FORMAT

LD [pathname...] [options]

LD lists the objects in a directory on standard output. It provides a wide variety of information on the contents of the various objects, depending on the command options that you select.

ARGUMENTS

pathname

(optional)

Specify pathname of object to be described. The object may be a directory, a file, or a link. If you specify a directory, LD describes the files in that directory. If you specify a file, the attributes of that file are reported. Multiple pathnames and wildcarding are permitted. (If they are used, each name is assumed to be a filename.)

Default if omitted: list contents of working directory

OPTIONS

Default options are indicated by "(D)."

Attributes

-A Display all attributes.

-ATTR Display permanent/immutable/trouble flags.

-BL Display disk blocks used.

-LEN Display current length in bytes.

-R Display your access rights to entries.

-ROOT Display the contents of the replicated root directory managed by

the naming server helper.

-ST Display system object type.

-TU Display type UIDs.

Date and Time

-D Display creation, modified, and last used dates.

-DTC Display date/time created.

-DTM Display date/time last modified.

LD (LIST_DIRECTORY)

-DTU Display date/time last used. Streams -SI Display all stream header information. -AB Display streams ASCII/binary flag. -CONC Display streams object concurrency. -RT Display streams record type. Entry Selection -AF d Display entries modified after date and time "d". -BE d Display entries modified before date and time "d". -DI Treat all names as directory names and list the contents of those directories. -EN Treat all names as entry names. -LD (D) List directory names. If this option is specified, then -LF and -LL lose their default status, and must be specified explicitly, if desired. -LF (D) List file names. If this option is specified, then -LD and -LL lose their default status, and must be specified explicitly, if desired. -LL (D) List link names. If this option is specified, then -LD and -LF lose their default status, and must be specified explicitly, if desired. -LT Display link resolution names. Output Control -C List entries in a single column, suppress header. (D) -HD Display header and totals. -NHD Suppress header and totals. -SN (D) Sort entries by name. -NSN Suppress entry sorting. -WARN (D) Produce a warning if no wildcard matches are found. Suppress warning if no wildcard matches are found. -NWARN LD uses the command line parser, and so also accepts the standard command options with the exception of the query options (-QA, -NQ, -QW).

TIME

The time at which a file is created, modified, or used is accurate within a certain tolerance. The reported time of creation or modification is correct within one minute of the actual creation or modification time. The time of last use is updated only if more than an hour has elapsed since the recorded time of last use. Hence the time of last use reported by the LD command may vary by as much as an hour from the actual time of last use.

EXAMPLES

1. \$ ld -a

Directory "/col/users/final1":

sys type	type uid	blocks used	current	attr	rights	name
file	rec	18	17640	P	pndwrx	ch1
file	rec	18	18428	P	pndwrx	ch2
file	rec	67	67210	P	pndwrx	ch3
file	rec	12	11554	P	pndwrx	ch4

4 entries, 115 blocks used.

2. \$ 1d -dtm

Directory "/col/users/final1":

date/t		
modifie	name	
82/03/28	17:18	ch1
82/03/28	17:18	ch2
82/03/28	17:19	ch3
82/03/28	17:20	ch4

4 entries, 115 blocks used.

3. \$ 1d /sys/ins/[a-e]?*.ins.ftn -a

sys type	type uid	blocks used	current length	attr	rights	name
file	rec	1	872	P	pndwrx	/sys/ins/base.ins.ftn
file	rec	2	1274	P	pndwrx	/sys/ins/cal.ins.ftn
file	uasc	20	19966	P	pndwrx	/sys/ins/core.ins.ftn
file	rec	1	738	P	pndwrx	/sys/ins/ec2.ins.ftn

4 entries listed, 24 blocks used.

LKOB (LOCK_OBJECT) -- Lock an object.

FORMAT

LKOB pathname [options]

LKOB locks the specified object. The locking constraint is "n readers XOR 1 writer".

LKOB is primarily used for system-level debugging.

To list locked objects, use LLKOB (LIST_LOCKED_OBJECTS). To unlock an object, use ULKOB (UNLOCK_OBJECT).

ARGUMENTS

pathname

(required)

Specify object to be locked. Multiple pathnames and wildcarding are permitted.

OPTIONS

Default options are indicated by "(D)."

-R (D) Lock the object for reading.

-W Lock the object for writing.

-I Lock the object for reading, with intent to write.

-R2W Change the lock mode of the object from "read" or

"read-intend-write" to "write".

-R2RIW Change the lock mode of the object from "read" to

"read-intend-write".

-W2R Change the lock mode of the object from "write" to "read".

-W2RIW Change the lock mode of the object from "write" to

"read-intend-write".

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ 1kob susan -w

Lock file "susan" for writing.

LLKOB (LIST_LOCKED_OBJECTS) -- List locked objects.

FORMAT

LLKOB [-R]

This command lists the locked objects resident on volumes mounted on this node, and objects resident in other nodes that are locked by processes running locally.

The listing for each object includes the locking constraints imposed on the object (e.g., n-readers XOR 1-writer), the specific lock mode being used (e.g., Read, Write, Read-Intending-Write), the network node ID of the node at which the object is located, the node ID of the node in which the locking process is active, and the name (if it is available) of the object itself.

OPTIONS

-R

Specify list of only those objects that either reside on this node and are locked by another node, or reside on another node and are locked by this node (i.e., those objects whose locks are in some way remote).

EXAMPLES

\$ 11kob

USE	CONSTRAINT	HOME NODE	LOCKING NODE	FILE
W	nR_xor_1W	21	21	/sys/dm/pdb
R	nR xor 1W	21	21	/sys/dm/fonts/std
W	nR xor 1W	21	21	Temporary File
R	nR xor 1W	21	21	Uncataloged Permanent File
W	nR_xor_1W	21	21	Display Manager Pad

LOGIN -- Log in to a running process.

FORMAT

LOGIN [person[.project[.org]] [-LP [passwd]] [-C pathname args...]]

The LOGIN command allows you to log in to a running process with a different identity. This permits you to have multiple concurrent processes running under different Subject Identifiers (SIDs). See the EDACL command description for an explanation of SIDs.

PLEASE NOTE that this command was designed to be invoked using the DM command 'cp /com/login', although it will also run in an existing Shell. You should be aware of two things when using this command:

- 1. The DM does NOT assume the new identity, just the process using the window does. Thus, activities performed by the DM are done using your original DM login identity. In particular, note that access to objects using the 'edit' (CE) and 'read' (CV) keys is determined by the DM's identity only.
- 2. The LOGIN command runs as a protected subsystem and requires a new process when it is invoked. If you log in to an existing window, a new process is spawned that shares the window but is not known to the DM. Thus, activities performed by the DM do not use the context of the new process but only the context of the original process. For example, the working directory used for the 'edit' and 'read' keys will be that of the original process, which is NOT necessarily the one shown if you type 'WD' in the window. Invoking LOGIN directly using 'cp /com/login' will remove the extra process layer and ease this problem. However, the DM will STILL not assume the process identity; it will merely be in direct contact with the new process.

To log out of a running process and return control of the process to the original SID, enter an end-of-file mark (usually CTRL/Z) in the process input pad.

ARGUMENTS

person (optional)

Specify your username. If you omit 'person', then LOGIN prompts you to log in interactively. In this case, respond just as you do to the DM log in prompt, i.e., issue the L command in the form:

L person[.project[.org]] [-P] [-H]

You will be asked for a password, and may take the opportunity to change your password and login home directory with the -P and -H options. See the L command description in the DM command chapter for more information. When you have successfully logged in, the process in the window assumes the new identity, and the node's local registry is updated.

Default if omitted: prompt for 'L' command

project
(optional)

Specify project ID (if one exists). This ID may be separated from 'person' by either a blank space or a period.

Default if omitted: no project ID specified

org

(optional)

Specify organization ID (if one exists). This ID may be separated from 'person' by either a blank space or a period.

Default if omitted: no organization ID specified

OPTIONS

-LP [passwd]

Specify password. If the 'person' argument appears on the command line, and -LP is not specified, you will be prompted for a password. If -LP is specified without an associated password, a blank password is used. -LP must follow the 'person' argument and precede the -C option. In addition, "-C" cannot be the password. Note that using -LP makes your password visible in the window.

After a successful login, the node local registry is updated with the new identity.

-C pathname [args ...]

Specify a program (followed by optional arguments to be passed to the program) that is to be invoked in the window after a successful login. If -C is not specified, /COM/SH (the Shell) is invoked. -C must not precede the 'person' argument.

EXAMPLES

\$ login

Please log in: 1 user

Password:

Logged in as user.none.none Monday, March 5, 1984 11:06:55 (EST). \$ args "And now for something completely different."

\$ (CTRL/Z) *** EOF ***

process stop (OS/fault handler)

Control returned to original SID

LOPSTR (LIST_OPEN_STREAMS)

LOPSTR (LIST_OPEN_STREAMS) -- List open streams.

FORMAT

LOPSTR

LOPSTR lists the streams that are open for the current process. The list contains the stream ID and access mode (read, write, append, and so forth) for each stream. The pathname (if one exists) associated with each stream is also displayed.

LOPSTR requires no arguments or options.

EXAMPLES

\$ lopstr

st# open name

O read (standard input)
1 append (standard output)
2 read (error input)
3 append (error output)

4 streams open.

LRGY (LIST_REGISTRY) -- List registry sites.

FORMAT

LRGY [options]

LRGY lists registry site names and the name of the network master registry file. For complete information on the use of local and network registries, see Administering Your DOMAIN System.

OPTIONS

If no options are specified, LRGY lists the sites in the current node's registry file copy (/REGISTRY/REGISTRY).

-R pathname

Specify name of registry file to be listed. If the -LOC option is also present, this name must be a node name (see example 3). If you omit this option but include -LOC, LRGY lists your node's local registry (//'node'/REGISTRY/LOCAL REGISTRY).

-LOC

List local registry.

EXAMPLES

- 2. \$ lrgy -loc List current node's local registry.

 Registry: //tape/registry/local_registry

 Sites of registration data files:
 //tape/registry/local_site

 Registry is LOCAL. It has 11 slots for login;
 the expiration period is 10 days.
- 3. \$ lrgy -r //os -loc List local registry of node "os".
 Registry: //os/registry/local_registry
 Sites of registration data files:
 //os/registry/local_site
 Registry is LOCAL. It has 11 slots for login;
 the expiration period is 10 days.

LTY (LIST_TYPES)

LTY (LIST_TYPES) -- List installed types.

FORMAT

LTY [volume_name] [options]

LTY lists the types currently installed on a volume. It can also be used to list the contents of internal caches for debugging purposes.

ARGUMENTS

volume_name

(optional)

Specify the volume for which types should be listed.

Default if omitted: list types installed on the boot volume.

OPTIONS

-U

Display type UIDs as well as type names.

-GLOB

Display contents of global type name cache instead of the type

file (for debugging only).

-PRIV

Display the contents of the private (per-user) type name cache

instead of the type file (for debugging only).

EXAMPLES

\$ lty

Local type file

area	bitmap	boot	casehm	ddf	evetype	hdru	ipad
lheap	mbx	mt	nil	null	obj	objlib	pad
pipe	rec	sch	sio	uasc	und	100.0	•

LUSR (LIST_USER) -- List logged on users.

FORMAT

LUSR [options]

LUSR lists the identities of active users on the network.

OPTIONS

If no options are specified, the person name and node entry directory of all users logged into the DM are listed.

-ME List the user logged on to this node by person, project,

organization name, and node ID.

-N pathname Lists user(s) logged on to the node specified. You may give a

hexadecimal node ID instead of a pathname if so desired. If the node ID starts with a letter, precede it with a zero (i.e., "0B3", not "B3"). Multiple pathnames or node IDs are permitted:

separate them with blanks.

-BR List only usernames.

-FULL List person, project, organization names, and node IDs.

-ALLP List identities for all user processes, not just the DM, either by

node (if -N is also specified), by name (-PPO), for the current

node only (-ME), or everywhere in the network.

-NOFULL List just the person name of each user listed.

-PPO ppo List user(s) named, at all nodes to which they have logged in to

the DM. 'ppo' is a string of the form 'pers.proj.org', where '%' may be used as a wildcard specifier and trailing %'s may be

omitted (e.g., %.os_dev or joe.%.r_d).

EXAMPLES

1. \$ lusr -me loc.none.mfg.1D5 //ET \$

2. \$ lusr -n //magic //mountain //park joe.none.mkt //MAGIC brian.none.none //MOUNTAIN gordon.hifi.none //PARK

LUSR (LISR_USER)

3. \$ lusr -full jtj.none.none.532 //zoid andy.none.now.12B //me carol.none.mtg.334 //vip *** diskless 383 *** partner node: //plan nelson.none.pres.838 annie.none.r_d.6CA //lunar now.system.advent.368 *** diskless 368 *** partner node: //zoid beth.none.mfg.2F7 //mack \$

LVAR (LIST_VARIABLES) -- List information about set variables.

FORMAT

LVAR [var_name ...]

The LVAR command lists the type, name, and value of currently set variables. Optionally, you can specify individual variable names.

ARGUMENTS

var_name ...

(optional)

List type, name, and value of the specified variable(s).

Default if omitted: list information for all variables currently set

LVOLFS (LIST_VOLUME_FREE_SPACE) -- List free space on logical volumes.

FORMAT

LVOLFS [pathname] [options]

LVOLFS prints information about the amount of available storage on mounted volumes. This information includes the total amount of storage in disk blocks, the amount of free storage, the percent of the total storage that is free, and the entry directory name for the volume.

ARGUMENTS

pathname

(optional)

Report on the volumes mounted on the home node of the

specified file.

Default if omitted: list free space on current node

OPTIONS

If no options are specified, LVOLFS reports the storage available on the volumes mounted on the current node.

-A

Report on all volumes mounted in the network.

-N node id...

Report on the volumes mounted on the specified node[s]. Multiple entry directories or hexadecimal node IDs are permitted; separate them with blanks.

EXAMPLES

\$ LVOLFS -A

free	total	% free	node id	entry directory
24217	30012	81	1A	/
16589	30012	55	2B	//DEV
7927	30012	26	ЗC	//LANG
14497	30012	48	4D	//MKT

MACRO -- Expand macro definitions.

FORMAT

MACRO [-0] [pathname ...]

MACRO is a general-purpose macro processor. MACRO reads the files and writes to standard output a new file with the macro definitions deleted and the macro references expanded.

ARGUMENTS

pathname

(optional)

Specify file containing macro definitions to be processed.

Multiple pathnames are permitted.

Default if omitted: read standard input

OPTIONS

-0

(Zero, not letter O) Remove one level of brackets in macro calls prior to processing. Normally, brackets appearing outside any macro calls (level zero brackets) are NOT removed.

A macro is a symbolic constant; when you use MACRO, each constant is replaced by the string of characters which define it. The general form of a macro definition is:

DEFINE(name, replacement text)

The string 'name' can consist of letters (a-z and A-Z), digits (0-9), underscores (_), and dollar signs (\$). All subsequent occurrences of the string 'name' separated from other letters, digits, underscores, and dollar signs by any other characters, spaces, or newline characters will be replaced by the replacement text. No space is allowed between the command (in this case, DEFINE), and the left parenthesis.

Blanks in definitions are significant; they should appear in the replacement text only where desired. Uppercase and lowercase letters are also significant. The replacement text may be more than one line long. However, when an entire macro definition is followed immediately by a newline, the newline is discarded. This prevents extraneous blank lines from appearing in the output.

A simple example of a macro is:

DEFINE (EOF, -1)

Thereafter, all occurrences of 'EOF' in the file would be replaced by '-1'.

You may specify arguments in macro definitions with the characters '\$n', where n is a number between 0 and 9. The arguments to be inserted when the macro is encountered are given inside parentheses following the macro name. \$0 refers to the name of the macro itself. For example,

```
DEFINE(copen,$3 = open($1,$2) )
```

defines a macro that, when called by

```
copen(name, READ, fd)
```

expands into

```
fd = open(name, READ)
```

If a macro definition refers to an argument that was not supplied, the \$n will be ignored. The \$ is taken literally if a character other than a digit follows it.

Macros can be nested, and can be called recursively. Any macros encountered during argument collection are expanded immediately, unless they are surrounded by square brackets ([]). That is, input surrounded by brackets is left absolutely alone, except that one level of [and] is stripped off. Thus it is possible to write the macro D as

```
DEFINE(D, [define($1,$2)])
```

The replacement text for D, protected by the brackets, is literally 'DEFINE(\$1,\$2)' so you could use:

```
D(a,bc)
```

to define a as bc. Brackets must also be used to redefine a macro. For example:

```
DEFINE(x,y)
```

```
DEFINE(x,z)
```

will define y in the second line, instead of redefining x. To define x the second time, the operation must be expressed as

```
DEFINE(x,y)
```

```
DEFINE([x],z)
```

Normally, brackets appearing outside any macro calls (level zero brackets) are not removed. When the -0 (zero, not letter O) option is specified, one level of brackets is removed both inside and outside the macros. One level of brackets is also removed when the macro reference is expanded. Thus, to rewrite the 'D' macro above so that it is evaluated to the literal string 'define(\$1,\$2)', the definition is:

```
DEFINE(D, [[define($1,$2)]])
```

In order to redefine the macro 'DEFINE' (for example, so that the Pascal keyword 'DEFINE' can be used) the following definition can be used:

DEFINE([DEFINE], [[DEFINE]])

Both arguments get one level of brackets stripped when the definition is processed; the second argument gets another level stripped when the macro is invoked.

The following built-in macros are provided:

DEFINE(a,b) defines a to be b and returns the null string.

IFELSE(a,b,c,d) returns c if a is identical to b. Otherwise, it returns d.

INCR(a) interprets a as an integer and returns a+1.

SUBSTR(a,m,n) returns a substring of the string a starting at character number m and extending for n characters.

LEN(a) returns the length of a.

INCLUD(a) returns the contents of file a.

EXPR(a) returns the result of evaluating infix expression a. Operators in increasing order of precedence are as follows. Parentheses may be used as usual.

DIAGNOSTICS

arith evaluation stack overflow

Arithmetic expressions can only be nested to 30 deep.

arg stack overflow

The total number of arguments exceeds the limit of 100.

call stack overflow

Definitions can only be nested to 20 deep.

EOF in string An end-of-file has been encountered before a bracketed string has been terminated.

evaluation stack overflow

Too many characters are used for the name, definition, and arguments of one macro. 2500 characters are allowed.

unexpected EOF An end-of-file was reached before a macro definition was terminated.

filename: can't open

The named file could not be opened.

filename: can't include

The indicated file cannot be included with the built-in macro INCLUD.

includes nested too deeply

Files included with the built-in macro INCLUD can be nested only up to 128 deep.

expression: invalid infix expression

There is a syntax error in the indicated infix expression as passed to the built-in macro EXPR.

too many characters pushed back

A macro expansion is too large to be rescanned. A macro definition may contain up to 2500 characters.

name: too many definitions

The table space for macro definitions has been exhausted; this occurred upon the definition of the named macro.

token too long A name or symbol in the input was longer than the token buffer. Each token may be up to 200 characters long.

MTVOL (MOUNT_VOLUME) -- Mount a logical volume.

FORMAT

MTVOL disk_type[unit] [log_vol_number] [pathname] [options]

A logical volume is a named storage area on a disk. MTVOL mounts a logical volume, making the files and directories it contains accessible. Up to eight volumes (both physical and logical) may be mounted on a node at any time. Of the eight, no more than five of those volumes may be logical.

Before a new physical volume can be mounted for the first time, you must initialize it. See the appendix on INVOL for details.

ARGUMENTS

disk_type (required)

Specify the type of disk on which the volume being mounted resides. Valid disk types are: W (Winchester), S (Storage Module), or F (Floppy).

unit (optional)

Specify disk unit number (0 or 1). If you use this argument, the unit number must follow the disk_type ID immediately: no blanks in between. For example, "S1" denotes storage module unit 1.

Default if omitted: 0

log_vol_number
(optional)

Specify the disk volume number. This is the same number that you assigned when you formatted the disk using INVOL. The first logical volume is numbered 1, the second 2, and so forth.

Default if omitted: 1

pathname (optional)

Specify the name of the volume entry directory. This is the logical volume's top-level directory. Specify this pathname only if the entry directory is not already cataloged in the naming tree. If the pathname you choose already exists, an error will result.

Logical volume entry directories may appear anywhere in the naming tree, with one exception: if a logical volume entry directory is also the node's entry (top-level) directory, it must appear just below the network root directory (//).

If you omit the pathname argument, MTVOL assumes that the entry directory already exists, and searches the naming tree for it. If it finds the entry directory, MTVOL mounts the volume and prints the full entry directory pathname.

If MTVOL does not find the entry directory, it prints an error message, and does not mount the volume. The search may fail for any of the following reasons:

- The entry directory has never been cataloged.
- The entry directory was uncataloged when the volume was last dismounted.
- The entry directory pathname exists on another node, for which directory information is currently unavailable.

An unsuccessful search does not mean that you cannot mount the volume. It simply means that the volume entry directory pathname does not exist on your node. To mount the volume, issue the MTVOL command and supply an entry directory pathname.

Even if the MTVOL finds the entry directory pathname, the mount may fail if the volume is corrupt for some reason and needs salvaging. In this case, MTVOL asks for permission to mount the volume. You should usually respond "no" to this request, then run the volume salvaging routine SALVOL. Once the volume has been salvaged, you may try to mount it again. If you mount a corrupt volume without salvaging it first, damage to files in that volume may result.

Default if omitted: see above

OPTIONS

Force -- Mount the volume whether or not it needs salvaging, and do not ask for permission.

-NQ No query -- Suppress query if a volume needs salvaging.

Instead, mount the volume ONLY if it does not need salvaging.

-PR Protect -- Mount the volume with write protection. Any attempts to write on the volume will fail.

CAUTION: Before removing a floppy disk volume mounted with MTVOL, you MUST use DMTVOL to dismount it. Failure to dismount the volume could result in lost or corrupt information.

EXAMPLES

\$ mtvol f Remount it using the new entry directory.

Volume mounted, entry directory is "/masterfloppy"

\$

MVF (MOVE_FILE) -- Move a file.

FORMAT

MVF source [destination] ... [options]

MVF moves a file to a different location in the naming tree. Its effect is identical to

\$ CPF source destination

copy source to destination

\$ DLF source

delete the source

MVF always retains the source ACL on objects moved.

ARGUMENTS

source

(required)

Specify name of file to be moved. Wildcarding is permitted.

destination

(optional)

Specify new file location. This pathname may be a derived name. If 'destination' is a directory, the command moves the source file into that directory. Otherwise it creates the new file using the name specified.

Default if omitted: copy source to current working directory

Multiple source/target pairs and wildcarding are permitted.

OPTIONS

Default options are indicated by "(D)."

-C (D) Create the target file. If the target file already exists, an error will result.

-R Replace target file with source file. Use this option if the target file already exists. If the file does not exist, this option works like -C.

Force deletion of destination object if you have 'p' (protect) rights, even if you do not have 'd' (delete) rights.

-LF List files moved.

-LDL List files deleted by -R option.

-CHN

Change the name of an existing destination file if required.

This option modifies the meaning of -C and -R. If -C is specified, this option causes any existing object with the destination pathname to be renamed prior to the move. If -R is specified, the destination object is renamed if it is in use and cannot be deleted.

MVF (MOVE_FILE)

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ mvf //anger/sam/letter //mary -r

Move the file "letter" from the directory "//anger/sam" to the directory "//mary" and replace the current file.

ND (NAMING_DIRECTORY) -- Set or display naming directory.

FORMAT

ND [pathname]

The ND command sets or displays the name of the naming directory. The naming directory is provided so that you may use a tilde (\sim) as a shorthand feature in pathname specifications. It is also important since the system checks its COM subdirectory (\sim COM) as a part of the default command search operation. The naming directory is set to the login home directory at login.

ARGUMENTS

pathname (optional)

Specify directory name to be used as the naming directory. ND also accepts the command line parser arguments "-" and "*". If you specify a hyphen (-), ND looks to standard input for the directory name. An asterisk (*) followed by the name of a file directs ND to look inside that file for the new naming directory name.

Default if omitted: display the name of the current naming directory.

EXAMPLES

\$ nd /paul/links Set naming directory to "/paul/links".

After execution of this command, you can use a tilde (~) in place of '/paul/links' at the beginning of any pathname. Thus "~sausage" would be the same as "/paul/links/sausage".

NETMAIN (NETWORK_MAINTENANCE) -- Analyze network maintenance stats.

FORMAT

NETMAIN [options]

NETMAIN is a highly interactive, menu-driven program that lets you control the NETMAIN_SRVR network maintenance server and analyze the data that NETMAIN_SRVR produces. NETMAIN provides detailed help from its menus. See Administering Your DOMAIN System for a complete description of NETMAIN's features, instructions about using its menus, and details about interpreting its output.

OPTIONS

Default options are indicated by "(D)."

-W[HELP] (D) Make sure window is large enough to display command menus and interactive help.

-WC[MD] Set window size smaller for command menus only. If you later decide that you want to see the helps, grow the window

manually with <GROW> or CTRL/G.

-NW Do not change window size.

EXAMPLES

1. \$ netmain Run NETMAIN in a window large enough to display command menus and interactive help.

2. \$ netmain -wc Run NETMAIN in a window large enough (but no larger) to display the command menus.

NETMAIN_CHKLOG (CHECK_NETMAIN_LOGS) -- Clean up bad log files.

FORMAT

NETMAIN_CHKLOG pathname... [options]

When the NETMAIN_SRVR program halts catastrophically (for instance, during a node reset), it can leave the log file it was writing in a corrupt, unusable state. NETMAIN_CHKLOG determines whether the log is corrupt and, optionally, deletes corrupt files.

If the pathname you specify points to some kind of file other than a NETMAIN log file, that file will almost always be ignored: it will almost never be deleted as a corrupt log. On very rare occasions, another kind of file may look so much like a corrupt log that it might be deleted accidentally if you use both -D and the standard command option -NQ (no query). Thus you should use -D -NQ with extreme care.

ARGUMENTS

pathname

(required)

Specify the files to be checked. Multiple names and wildcarding are permitted; separate names with blanks.

OPTIONS

Default options are indicated by "(D)."

-D

Delete corrupt log files.

-ND

(D) Do NOT delete anything.

-L

(D) Describe every file analyzed.

-NL

Describe only corrupt log files.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ netmain chklog 'node_data/net_log/?*

NETMAIN_NOTE -- Place message in network error log.

FORMAT

NETMAIN_NOTE string [string ...]

NETMAIN_NOTE sends a text string to NETMAIN_SRVR, the network maintenance server. The message is broadcast to all maintenance servers.

Typical topics of maintenance notes include known or explainable network failures, scheduled down-time, and node installations.

ARGUMENTS

string

(required)

Specify message to be sent. You may use any string that is legal in a Shell command. (Note that the Shell takes special action on some keywords, such as 'if', unless you place them in quotes.) If there is more than one string, NETMAIN builds the note by concatenating the arguments that are separated by spaces.

EXAMPLES

- \$ NETMAIN NOTE 'Scheduled down time at 5 pm.'
- \$ NETMAIN_NOTE Cable disconnected at //sancho_panza

NETSTAT (NETWORK STATISTICS) -- Display network statistics.

FORMAT

NETSTAT [options]

This command writes a summary of network and hard disk activity to standard output.

OPTIONS

If no options are specified, NETSTAT returns a brief summary of network usage information for the current node.

L Long Report -- Provide more information than the summary.

-CONFIG Configuration Report -- Display only node-specific hardware

information: cpu type, display type, etc.

-N node-spec Provide information on specified node(s). Multiple hexadecimal

node IDs are permitted, separated by blanks. If the node ID starts with a letter, precede it with a zero (i.e., "0B3", not "B3"). Node-spec may alternatively be the pathname of any

object on the remote node (i.e., //that_node).

-A Report on all nodes in the network.

-R [n] Repeat NETSTAT command every n seconds until halted by

CTRL/Q. Only counts that have changed at each iteration are displayed, and the values represent the amount of change rather

than absolute values. The default value for 'n' is 10 seconds.

-S [n] Send 'n' test messages to every node being listed (except the

current node) before every repeat of the display. If this option is specified, -R must also be specified. This option provides a minimum amount of network activity during the wait time between NETSTAT repeats. The default value for 'n' is 100

messages.

-SAVE pathname

Save all statistics in file named 'pathname'.

-SINCE pathname

Display counts that have changed since statistics were saved in

'pathname'.

EXAMPLES

1. \$ netstat

The node ID of this node is 1FB.

**** Node 1FB **** "//anger"

Up since 1983/02/01 at 8:17:06 Up for 1 day 2 hours 58 mins 4 secs

Net I/O: total= 94625 rcvs = 66912 xmits = 27713
Winchester I/O: total= 0 reads= 0 writes= 0 {NOTE 1}
System configured with 1.0 mb of memory.

2. \$ netstat -L

The node ID of this node is 1FB.

**** Node 1FB **** "//anger"

10436 page-in requests issued. 6473 page-out requests issued. 41134 page-in requests serviced. 12139 page-out requests serviced.

Detected concurrency violations -- read: 0 write: 0

Xmit count	27746		Rcv	eor	0
NACKs	272		Rcv	crc	767
WACKs	1639		Rcv	timout	0
Token inserted	65		Rcv	buserr	0
Xmit overrun	0	100	Rcv	overrun	0
Xmit Ack par	3		Rev	xmit-err	3042
Xmit Bus error	0	1	Rcv	Modem err	0
Xmit timout	90		Rcv	Pkt error	45
Xmit Modem err	0		Rcv	hdr chksum	0
Xmit Pkt error	377	1	Rcv	Ack par	10

Delay switched OUT.

Winchester I/O:	total=	0	reads= 0	writes=	0	{NOTE 1}
Not ready	0		Contrlr busy	y 0		
Seek error	0		Equip check	0		
Drive time out	0		Overrun	0		
CRC error percen	ntage: 0.0	00%				

Last ring hardware failure detected by node 241 on 1983/02/02 at 10:05
System configured with 1.0 mb of memory.
A total of 0 ECCC errors were detected.

{NOTE 2}

Notes on Examples

- 1. Node 1FB is running diskless, hence the absence of Winchester disk I/O activity.
- 2. At 10:05 A.M. on Feb. 2, 1983, the network cable was disturbed immediately upstream of node 241. This information, coupled with the network topology available from LCNODE (LIST_CONNECTED_NODES), can help you pinpoint a hardware malfunction.

NETSVC (NETWORK SERVICE) -- Set or display network services.

FORMAT

NETSVC [options]

NETSVC sets or displays the network services that this node will perform. All changes take place immediately.

OPTIONS

If no options are specified, NETSVC displays the network services allowed for this node.

Default options are indicated by "(D)."

None -- Disable all network services and physically disconnect this node from the network.

Local -- Allow only network requests originating at this node.

-R Remote -- Allow only network requests originating at other nodes.

-A (D) All -- Allow both locally and remotely initiated network requests. (The size of the remote paging pool is not changed.)

-S[ERVERS] [n]

Servers -- Set the number of network servers to run on this node. At system statup, the number of network servers is 1. If this node is a network partner for diskless nodes or has several remote file users, their performance could be improved by increasing the number of servers. If n is not specified, the maximum number of servers (5) is used.

-P [n] Pool -- Set local memory pool size. Network page requests originating at remote nodes may not use more than 'n' pages of the local node's memory. If n is not specified, all of the local node's memory is eligible for remote page requests.

CAUTION: Be careful when revoking network access with -N or -L. Remote file users may have problems, and writable files may be damaged. If your node was the network partner for a diskless node, that node will crash when your node leaves the network.

The -N option disallows ordinary network requests to or from your node and mechanically disconnects your node from the network. Once you have used the -N option, some network service or analysis tools (MBX_HELPER, PROBENET, NETSTAT's -S option, and others) can reconnect your node with the network. When your node is reconnected by these programs, it still ignores ordinary requests for network service. Use the -L , -R, or -A option to allow more normal access to the network.

NETSVC (NETWORK_SERVICE)

Use the -S option carefully. Although you can increase the number of servers, you cannot decrease it. The only way to return to a smaller number of servers is to reboot the node. Also note that increasing the number of server processes decreases the number of user processes allowed. When you run NETMAN, the number of servers increases from one to two. If the number of servers is already two or greater when you start NETMAN, it will not increase further.

EXAMPLES

\$ netsvc
Network operations allowed: ALL
Number of network servers: 1
Remotely initiated paging pool limit: NONE
\$

NEXT -- Return to the top of a loop.

FORMAT

NEXT

NEXT interrupts the flow of control in a Shell loop construct (FOR, SELECT, and WHILE). When NEXT is encountered in a FOR or WHILE loop, control passes back to the top of the loop (see examples below). When NEXT is encountered in a SELECT loop, control passes to the next CASE clause. (This is useful when you have specified SELECT ONEOF but want to test multiple things under certain circumstances).

You may terminate the flow of control in a loop by using the EXIT command. See the EXIT command description for more information.

The NEXT command requires no arguments or options.

EXAMPLES

Consider the following section from a Shell script:

```
n := 0
WHILE ((^n < 10))
DO     READ -TYPE integer n
     IF ((^n < 10)) THEN NEXT ENDIF
     ARGS ^n
ENDDO</pre>
```

AS long as the READ command reads integers into variable "n" that are less than 10, the NEXT command executes and causes the script to return to the top of the WHILE loop. When the value of n is greater than or equal to 10, the script prints the number then leaves the WHILE loop and continues execution.

For more information on variables, refer to the DOMAIN System User's Guide.

NOT -- Negate a Boolean value.

FORMAT

NOT command

NOT takes the Boolean value returned by a command or expression and negates it. This is useful primarily with the program control structures (IF, WHILE, etc.) used in Shell scripts.

ARGUMENTS

command

(required)

Specify a command or expression that returns a Boolean value.

EXAMPLES

Assume the following lines appear inside Shell scripts.

```
#
# Loop as long as no error file exists.
#
while not existf error_file
do args "No error file yet ..."
enddo
# End of script
```

```
#
# Verify user response.
#
eon
read -p "Type the pathname of the file to be deleted: " name
read -p "Are you sure you want to delete ^name?" verification
if not ((^verification = "yes")) then
    read -p "Please retype the pathname: " newname
    dlf ^newname
else dlf ^name
endif
#
```

OBTY (OBJECT_TYPE) -- Set or display the type of an object.

FORMAT

OBTY pathname... [object_type]

OBTY is intended for system-level debugging use only. Misuse of this command can cause objects to become inaccessible and programs to behave incorrectly.

ARGUMENTS

pathname

(required)

Specify object whose type is to be set or displayed. Wildcarding of this pathname is permitted.

object_type

(optional)

Specify new type setting. 'Object_type' must be one of the following:

NIL (nil type UID)

UASC unstructured ASCII (text) file

REC streams records file

HDRU streams header-undefined file

OBJ object file

PAD display manager pad SIO sio descriptor file

UNDEF undefined file type NULLDEV null device (bit bucket)

DDF device descriptor file (for GPIO)

MBX mailbox

AREA D3M area file

SCH D3M object (sub)schema file MT magnetic tape descriptor file

BOOT system bootstrap file

Boot System bootstrap Tile

Executable files (output of compilers and binders) are OBJ. Most other binary files are REC.

Default if omitted: display current type of 'pathname'

OPTIONS

OBTY accepts no special options of its own, although it does use the command line parser, and so accepts the standard command options listed in Chapter 3.

EXAMPLES

The sequence of the following commands is significant.

\$ obty testfile

Display current object type.

"testfile" object type is nil.

\$ obty testfile uasc

Set type to "uasc".

OBTY (OBJECT_TYPE)

\$ obty testfile
"testfile" object type is uasc.

Display new object type.

OLD_EDFONT -- Edit a character font.

FORMAT

OLD_EDFONT

OLD_EDFONT is an interactive, menu-driven program that allows you to create, edit and view character font files. Its user interface is based on function-key menus, in contrast to the newer pointing device-driven menus of EDFONT. For a detailed explanation on editing a character font, see the OLD_EDFONT appendix.

OLD_EDFONT requires no arguments or options to begin processing.

OS (OVERSTRIKE) -- Convert ASCII to FORTRAN carriage control.

FORMAT

OS [pathname...]

OS converts a file containing ASCII carriage control (for such things as form feeds and backspacing for underlining) into a file that can be printed on a line printer with FORTRAN carriage control. By default, output is written to standard output; redirect it into a file with the ">pathname" expression.

If you create a new file containing the overstruck text, OS automatically sets the file's carriage control.flag so that printers we supply will interpret the file correctly. If you use OS in a pipeline, however, the flag is not set (since output goes to standard output). In this case, you must use the -FTN option on the PRF command for the file to be printed correctly. See examples 2 and 3 below.

ARGUMENTS

pathname

(optional)

Specify file to be converted. Multiple pathnames are permitted, separated by blanks. All output is concatenated, however.

needed.

Default if omitted: read standard input

EXAMPLES

1.	\$ os mary	Convert the file "mary" and write to standard output.
2.	<pre>fmt letter os >letter.overstruck prf letter.os</pre>	Format the file "letter", pipe output to OS, and write the results into "letter os." This file is then printed on the default printer.
3.	\$ fmt letter os prf -npag -ftn	Format the file "letter" and pipe it directly to the line printer. Note the use of "-ftn" to ensure that proper carriage control is used.
4.	\$ fmt letter prf -npag -pr spin	Format "letter" and print it on a Spinwriter printer. Since Spinwriters use ASCII carriage control, OS and the -FTN option on PRF are not

PAGF (PAGINATE_FILE) -- Paginate a file.

FORMAT

PAGF [pathname...] [options]

PAGF paginates the named files to standard output. Each file is printed as a sequence of pages. Each page is 66 lines long by default, including a 6-line header and 3-line footer. The header includes the file name, the date and time, and the page number.

ARGUMENTS

pathname

(optional)

Specify file to be formatted. Multiple pathnames are permitted

separated by blanks.

Default if omitted: read standard input

OPTIONS

-L n

Set the page length to 'n' lines. The default page length is 66

lines.

EXAMPLES

\$ pagf mary -L 20 >mary.short

Paginate the file "mary" into pages 20 lines long and write them to "mary.short"

PPRI (PROCESS_PRIORITY) -- Set or display process priority.

FORMAT

PPRI [process_name...] [options]

The process priority is an integer ranging from 1 (low) to 16 (high). When the operating system decides which process to run next, it chooses the process that currently has the highest priority. As a process executes, its priority increases as it waits for events (such as keyboard input) and decreases as it computes for long periods without waiting. By default, the priority is bounded by the range 3 through 14 when a process is created. The PPRI command lets you change these bounds to any other numbers in the range of 1 to 16.

A forked process inherits the priority settings of its parent process.

ARGUMENTS

process_name...

(optional)

Specify name of process whose priority is to be set or displayed. Multiple names and wildcarding are permitted.

Default if omitted: use current process.

OPTIONS

If no options are specified, the current priority bounds are displayed.

-LO n

Set priority lower boundary. 'n' must be in the range 1-16 inclusive. If this option is omitted, the lower boundary is set to 3.

-HI n

Set priority upper boundary. 'n' must be in the range 1-16 inclusive. If this option is omitted, the upper boundary is set to 14

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

- 1. \$ ppri Display defaults for current process MY_SHELL: minimum priority = 3, maximum priority = 14
- 2. \$ ppri process_7 -lo 1 -hi 4 Restrict process 7 to low priorities
- 3. \$ ppri -lo 12 -hi 12 Current process will always have priority 12

PRF (PRINT_FILE) -- Queue a file for printing.

FORMAT

PRF [pathname...] [options]

PRF queues a file for printing. The file must be an ASCII stream (i.e., text) file, a graphics metafile (GMF), or a GPR bitmap object. After successfully queueing a file, PRF displays a message containing the full pathname of the file that you queued.

You can execute PRF once for each file that you want to print (specifying all the necessary options every time), or you can enter PRF's interactive mode and hand files to the program continuously. See the examples below.

Files queued by PRF are physically printed using PRSVR (PRINT SERVER).

When you invoke PRF, it first sets all options to their default states (as described below). Next, it looks for a PRF configuration file called ~USER_DATA/PRF.DB unless you have invoked PRF with the -NDB option (described below). If PRF locates a configuration file, it executes the options contained in the file to configure the current session. Finally, it proceeds to process any options on the command line or in the interactive session.

A menu-based version of the PRF command is also available. See the PRFD command description for more information.

A Word About Printronix Carriage Control

The Printronix line printer that we supply can interpret two kinds of carriage control: ASCII (the default for DOMAIN files) and FORTRAN. In files with ASCII carriage control, the NEWLINE and FORM FEED characters have special meanings. NEWLINE terminates a line. FORM FEED causes a page eject if it is the only character in a record. In any other position, FORM FEED causes unpredictable results.

If a file has FORTRAN carriage control (which consists of special characters in the first column of every line in the file), PRF works one of two ways. If the file has been produced by the OS (OVERSTRIKE) command or by one of your own programs that explicitly sets the file's FORTRAN carriage control flag, PRF automatically treats every record as a line, and interprets the first character in the record as a carriage control character with its standard FORTRAN meaning. Within a program, you can set the carriage control output flag by calling STREAM_\$REDEFINE. To generate a file with FORTRAN carriage control from a file formatted with FMT, use the OS command.

If you know that the file has FORTRAN carriage control, but aren't sure whether or not the file's carriage control flag has been properly set, specify -FTN (below) to force use of this feature.

ARGUMENTS

pathname

(optional)

Specify the file to be printed. Multiple pathnames and

pathname wildcarding are permitted.

Default if omitted: read standard input.

OPTIONS

The following options may appear on the Shell command line or in PRF interactive mode as noted below. In addition, you may place one or more options in a configuration file (see -CONFIG). In that case, create the file with one option per line without the prepended hyphens (-). See Example 3 below.

If no options are specified, the file(s) are printed using ASCII carriage control, with pagination enabled, on the default printer (as established by PRSVR).

The following options apply to all file types.

-INTER[ACTIVE]

Enter interactive mode.

-COP[IES] n

Print multiple copies of the file, where 'n' is the requested number of copies. If -COP[IES] is specified, 'n' is required. If this option is omitted, one copy is printed by default.

-PR[INTER] name

Specify the printer 'name' for printing the file. This option is useful only if more than one printer is in use on the network, or if a printer has been assigned a nonstandard name with the "PRINTER_NAME" configuration directive in the PRSVR command. If you omit this option, PRF uses the default printer name, "P". Note that "P" is also the default printer name used by the PRINT_SERVER.

-S entry_dir

Specify print queue (/SYS/PRINT) on alternate node by giving that node's entry directory name. This option allows you to maintain more than one printer queue directory. You may want to maintain separate queues for different organizations, or you may want two queues to provide redundancy in case of node failure.

-USER[NAME] name

Specify user name that will appear on the banner page of the printed file. The alarm facility of PRF also uses this name to determine who should be notified when printing is complete (see -SIG below). This means that this name must be a valid login name (unless you don't care about sending an alarm). If this option is omitted, the current login name is used.

-SIG[NAL] ALARM|OFF

Request an alarm server signal when the file has finished printing. The default is OFF.

-BAN[NER] [ON|OFF]

Enable/disable banner page. The default is specified in the PRSVR configuration file. If neither ON nor OFF is specified, ON is assumed.

-CONFIG[_FILE] [pathname]

Specify a file containing further PRF options, one per line. Do not use prepended hyphens (-) with the option names in the configuration file. If 'pathname' is omitted, PRF will execute the configuration file ~USER_DATA/PRF.DB.

-NDB

Suppress processing of the configuration file.

-TEXT

Specify text mode for printing ASCII files. This is the default print mode.

pr.

-PLOT

Specify plot mode. Include this option to print bitmap files created by a graphics metafile (GMF) manager or GPR or the CPSCR (COPY_SCREEN) command.

-TRANSPARENT

Specify that when the file is printed, the records of the file are to be passed directly to the printer driver routine with no processing by the PRINT SERVER.

The following options apply to text files only.

-NPAG

Disable the headers and margins generated by PRF.

-MARGINS [ON|OFF]

Enable/disable margins generated by PRF. The default is 'ON'. If neither ON nor OFF is specified, ON is assumed.

-TOP n

Specify page top margin, in inches. The default is a value specified in the PRSVR configuration file.

-BOT[TOM] n

Specify page bottom margin, in inches. The default is a value specified in the PRSVR configuration file.

-RIGHT n

Specify page right margin, in inches. The default is 0 inches.

-LEFT n

Specify page left margin, in inches. The default is 0 inches.

-HEADERS [ON|OFF]

Enable/disable page headers and footers generated by PRF. The default is specified in the PRSVR configuration file. If neither ON nor OFF is specified, ON is assumed.

-HEAD[_STRING] l-string/c-string/r-string

Specify contents of left, center, and right components of the page header generated by PRF. Components may be empty strings. The following special characters return the values indicated when they appear in the header strings. @ = escape character

= current page number with 1 leading and
1 trailing space

% = current date

! = filename

& = filename's last time, date modified

* = insert a space in text string (literal spaces
are not allowed)

Example: -HEAD !/Page#/% will produce a header with the filename in the left component, the string "Page" followed by the current page number in the center component, and the current date in the right component. The default header is a string specified in the PRSVR configuration file.

-FOOT[_STRING] l-string/c-string/r-string

Specify contents of page footers. The format is the same as for -HEAD above. There is no default footer.

-FTN [ON|OFF]

Force use of FORTRAN carriage control. The -FTN option causes the PRINT_SERVER to use FORTRAN forms control even if the file does not have the FORTRAN carriage control flag. Use of this option will cause PRF to interpret the first character of each line as a FORTRAN carriage control character (and not print it). This can be unfortunate if the file has ASCII carriage control, so be careful. If neither ON nor OFF is specified, ON is assumed. The default state is OFF.

-WRAP [ON|OFF]

Enable/disable automatic line wrapping. When enabled, PRF will wrap any lines that exceed the right margin onto the next line. When disabled, PRF truncates lines that exceed the right margin. If neither ON nor OFF is specified, ON is assumed.

The following options are for use with printers supporting variable font and pitch sizes.

-PITCH n

Set the pitch (characters/inch) at which you wish the document to be printed. The following pitch settings are available on the printers indicated.

Printronix 10 Spinwriter 12

Imagen 8.5, 10, 12, 15, 17.1 GE 10, 12, 13.1, 16.7

Versatec 1

-POINT n

Set the point size for the font to be used. This is a real number in units of a point which is 1/72 inch.

-WEIGHT value

Set the weight of the font to be used. This option is only valid for the GE printer type. Possible values are 'light', 'medium', and 'bold'. The default is 'medium'. -LQ [ON|OFF]

Specify that the document is to be printed in 'letter quality' (ON) as opposed to 'draft' (OFF) mode. This option is only valid for the GE printer type. If neither ON nor OFF is specified, ON is assumed. The default state is OFF.

The following options apply to plot files.

-RES[OLUTION] n

Specify resolution of output plot in dots per inch. If you specify a resolution not available on the particular printer, the file is printed at the closest available resolution. The default resolution is specified in the PRSVR configuration file.

-WHITE[_SPACE] n

Specify amount of white space (in inches) to appear between multiple plots in one file. The default is three inches.

-BW[_REV] [ON|OFF]

Enable/disable black and white reversal for bitmaps. If neither ON nor OFF is specified, ON is assumed. The default state is OFF.

-MAGN[IFICATION] n

Specify bitmap magnification value. 'n' is an integer in the range -1 to 16.

-1 selects auto-scaling to magnify the bitmap to fill the available page space.

selects 'one-to-one' scaling between the display and the printer for GMF bitmaps. (For GPR bitmaps, this translates to magnification 1.)

1-16 selects magnification by that amount.

Portions of the magnified bitmap that exceed the printer page boundaries are clipped.

The default is 0.

0

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

COMMANDS

Once PRF has been invoked in interactive mode (see -INTER above), it accepts the following interactive commands at the "PRF> " prompt (in addition to the options already discussed).

P[RINT] pathname Queue the specified file for printing.

Q[UIT] Quit interactive mode and return to the Shell.

SH[ELL] Create a shell command line. This command allows you

to issue Shell commands without leaving PRF interactive mode. When you have finished entering Shell commands, type CTRL/Z. This will return you to PRF interactive mode. Your previous PRF option settings remain undisturbed by the intervening Shell commands.

INIT[LALIZE]

Reset PRF parameters to their default values.

R[EAD] [printer]

List entries in the queue for the specified printer at the current site (as specified by -S). If 'printer' is omitted, then the contents of the queue (determined by the current setting of -PR) are listed.

WD [pathname]

Execute the Shell command WD (WORKING_DIRECTORY) to set or display the working directory.

GET option

Display the value of the PRF option specified. Use this command to show the settings of the various PRF parameters.

lands -

CAN[CEL] [queued_filename]

Cancel printing of the specified file at the current site (as specified by -S). Note that you must specify the pathname which PRF assigns when the file is queued (which may differ from the name of your original file). Use the READ command to display the names of currently queued files. If the filename is omitted, the last file to be queued by this process is cancelled. This command is only effective for files which have not yet physically begun to print.

EXAMPLES

1. \$ prf mary -npag -ftn
 "//NODE1/MY_DIR/MARY" queued for printing.
\$

Queue "mary"; suppress pagination; force FORTRAN carriage

- 3. Configuration File: the following commands might appear in the default PRF configuration file "USER_DATA/PRF.DB.

PR ge SITE //rye FOOT %/my_file/&

4. Sample interactive session:

```
$ PRF -INTER
PRF> get pr
pr = p
PRF> -pr cx
PRF> get pr
pr = cx
PRF> -pitch 20
PRF> print test_file.pas
"//NODE1/MY_DIR/TEST_FILE.PAS" queued for printing.
PRF> q
$
```

5. Running PRF from an icon:

If you would like to run PRF interactively in a process devoted to it, you might place the following command in your "USER_DATA/STARTUP_DM file:

This will create a PRF process and turn its window into an icon using the print icon character in (/SYS/DM/FONTS/ICONS). Issue the DM command ICON to change the icon window into its full-size format.

PRFD (PRF_DISPLAY) -- Invoke menu-based PRF.

FORMAT

PRFD

PRFD invokes the menu-based version of the PRF (PRINT_FILE) command. This interactive version runs a graphics interface inside the process window in which PRFD was invoked. You control the operation of PRFD by pointing the cursor at a desired menu item and pressing <M1> (if you have a mouse) or <F1> or the space bar if you do not have a mouse. Internal help is available by pointing at a menu item and pressing the <HELP> key (R6S).

PRFD requires no arguments or options on the Shell command line. To exit PRFD and return to the Shell, select the "QUIT" menu item.

PRFD performs exactly the same functions as the command-line based PRF. See the PRF command description for complete information on those various functions.

PROBENET (PROBE_NETWORK) -- Probe network and display error statistics.

FORMAT

PROBENET [options]

This command broadcasts packets to the diagnostic socket in all nodes, then requests error counts indicating the status the broadcast was received with. It compiles counts from every node in the topology list and reports them to standard output.

OPTIONS

Default options are indicated by "(D)."

Use one of the following three options to specify the list of nodes to display.

-A (D) Probe all nodes responding to an LCNODE command. If the network is completely corrupted so that messages cannot make a complete pass, use one of the other two options to specify precisely which nodes to test.

-T pathname Probe the nodes listed in the topology file indicated. The file must contain one hexadecimal node ID per line. Any text following a space after the node ID is ignored. Comment lines may be inserted if they are prefixed with a '#' or '{'.

-N node_id ...

Probe the node(s) specified by the indicated hexadecimal node

ID(s). A good choice of nodes to test is a set evenly spaced

around the network.

Use the following options to specify which test to run.

-S n (D) Specify the total number of packets to be sent to each node.

The default number of packets is 10. If 0 is specified for 'n', no
test messages are sent, but statistics from each node will be
collected.

-R [n] Repeat PROBENET cycle every 'n' seconds. If 'n' is omitted, the cycle is repeated every 10 seconds. When <RETURN> is typed at the input window, the send cycle is terminated immediately and the statistics are gathered and reported.

Use the following options to specify which packets are sent.

-D data_file Specifies that the packets will be taken out of the specified data file instead of the standard built-in data pattern.

-LEN n (D) Specify the length (in bytes) of the data portion of the test packet, in bytes. The default length is 1024 bytes.

Use the following options to control the level of detail in the statistics report.

-L Print long (detailed) error counts if there were any errors (i.e., at least one XMIT ERRS or RCV ERRS).

PROBENET (PROBE_NETWORK)

-ERR

Print header for each test, but only statistics for nodes which returned errors (XMIT and/or RCV ERRS).

-MON fail lim

Print header for each pass, but only statistics on passes whose total failure count equal or exceed the 'fail lim' value.

-SENS threshold

Open a window pane and select some output lines to append to this pane. The nodes selected are the ones whose error count exceed a five node running average error count by the specified threshold value. Also, all nodes with modem errors are appended to this pane. The use of this secondary output is to do some data reduction and pick the nodes at or near points of data corruption in the network. The window pane is also stored in a named pad file: PROBENET.PANE

REPORTED STATISTICS

The following statistics are printed for each node:

ATTEMPT Number of probenet packets received.

ERRS Number of probenet packets received with errors. An increase in this count over previous nodes narrows the network problem between this

and the previously displayed node.

MODEM ERRS Number of transmit or receive modem errors encountered by the node.

BIPH Number of transmit or receive biphase errors encountered by the node.

An increase in this count over previous nodes narrows the network problem between this and the preceeding node, independent of probenet

display.

ESB Number of transmit or receive elastic store buffer errors encountered

by the node.

TOKENS Number of tokens inserted by this node. This statistic does not localize

any problem.

EXAMPLES

1. \$ probenet {Probe the entire network once. No errors detected.} There are 4 nodes in the test.

Broadcasting 10 1024-byte packets . 85/02/20 21:16:52 # failures = 0

Last Biph hardware failure detected by node 676 on 85/02/20 at 19:15

4				MODEM					
NODE	NAME	ATTEMPT	ERRS	ERRS	BIPH	ESB	TOKENS=	=	0
584	*diskless	10	0	0	0	0	0	Self	
AEF	BS	10	0	0	0	0	0		
4A	HUBRIS	10	0	0	0	0	0		
3536	*diskless	10	0	0	0	0	0		

- 2. \$ probenet -t node_list -s 14400 -r 3600 -d data e3
 - { Probes network and displays nodes specified in file "node_list". This node broadcasts 14400 packets in 3600 seconds, i.e. four packets per second. The packet data comes out of file "data e3".}

There are 5 nodes in the test.

Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds.

85/02/20 21:58:19 # failures = 100

Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

				MODEM					
NODE	NAME	ATTEMPT	ERRS	ERRS	BIPH	ESB	TOKENS	=	3
		-							
1967	GTX	14386	0	0	0	0	1	Self	
15F5	SWI	14386	0	0	0)	0		
2255	BIRDIE	14384	0	0	0	0	1		
3FD	FLASH	14386	3	3	3	0	0		
2B69	STANG	14385	3	0	0	0	1		

Broadcasting 14400 1024-byte packets (page 0) over 3600 seconds. 85/02/20 21:58:41 # failures = 100 Last Biph hardware failure detected by node 506 on 85/02/20 at 21:50

				MODEM				
NODE	NAME	ATTEMPT	ERRS	ERRS	BIPH	ESB	TOKENS=	3
1967	GTX	14383	0	0	0	0	1 Self	
15F5	SWI	14383	0	0	0	0	0	
2255	BIRDIE	14381	0	0	0	0	1	
3FD	FLASH	14383	4	4	4	0	0	
2B69	STANG	14382	4	0	0	0	1	

[{] Above example shows a problem between node 3FD and its predecessor in the network. }

PRSVR (PRINT_SERVER) -- Start the Print Server.

FORMAT

PRSVR [config_file_name] [options] [&]

PRSVR executes the print server program, which prints files submitted to the print queue with PRF (PRINT_FILE). You only need to execute this command when you start the node connected to the printer. Do not execute the command at other nodes: this will cause print files to be lost.

The print server must run independently of user login, or it will stop when you log out. To start the print server, include this command line

CPS /COM/PRSVR [pathname]

in the printer node's STARTUP file.

PRSVR first performs some internal initialization, then, after about ten seconds, prints an introductory message on the printer. It then scans the print queue (/SYS/PRINT/QUEUE), and if it finds a file intended for it, prints the file. After printing all the files in the queue, it checks the queue for more files every 10 seconds.

For complete details about user-supplied device drivers for auxiliary devices and the contents of the print server configuration file, see Administering Your DOMAIN System.

ARGUMENTS

pathname

(optional)

Specify printer configuration file.

Default if omitted: use file PRINTER_CONFIG.DATA.

&

(optional)

Create separate Shell process in which to run the print server. This process is created without the normal pads or windows, thus running invisibly in the background. When the print server is started in this fashion, it stops automatically at logout. To stop this background process before logout, use the SIGP (SIGNAL_PROCESS) command.

OPTIONS

-N name

Specify a name for the print server process. If this option is not specified, the process is named PRINT_SERVER.printername.

PST (PROCESS_STATUS) -- List process internal state information.

FORMAT

PST [options]

PST lists internal state information for all processes in the system by name or UID.

OPTIONS

-	\mathbf{R}	n				
---	--------------	---	--	--	--	--

Repeat every n seconds. If you include this option, the first pass displays the total time elapsed since process creation. Subsequent passes display changes from the previous pass, as shown in the example below.

-N node-spec

Specify remote node whose process stats are to be listed. Node-spec may be the target node's hexadecimal node ID, or the entry name of any volume on that node.

-UN

Display DOMAIN/IX process IDs.

-PA

Display process paging information. The paging data presented is private page faults, global page faults, disk paging I/O, and network paging I/O.

EXAMPLES

1. \$ pst -r 20

273.345 16/16/16 18088 Wait display: 441.379 1/15/16 <active> Ready process 68.086 1/16/16 17E90 Wait digital</active>	
68.086 1/16/16 17E90 Wait digital	
	1
	clock
7.531 1/14/16 17E90 Wait alarmer	
0.715 1/15/16 17C30 Wait mbx help	er
0.715 1/15/16 17F78 Wait mail	
0.715 1/16/16 17F78 Wait shell	
792.489	

Time	(sec)	mn/cu/mx	Counter		Name
	0.262	16/16/16	18088	Ready	display manager
	0.262	1/16/16	<active></active>	Ready	process 1
	0.262	1/15/16	17E90	Wait	digital clock
	0.000	1/14/16	17E90	Wait	alarmer
	0.000	1/15/16	17C30	Wait	mbx_helper
	0.000	1/15/16	17F78	Wait	mail
	0.000	1/16/16	17F78	Wait	shell
	0.786				

```
Display stats of node "//eve".
2. $ pst -n //eve
   Processor | PRIORITY | Program | State | Process
   Time (sec) | mn/cu/mx | Counter | Name
    1016.261 16/16/16 18088 Wait display_manager
       7.269 1/2/16 17C30 Wait netman
4.385 1/15/16 17C30 Wait mbx_helper
      10.939 1/16/16 17F78 Wait server_process_manager
    1038.855
3. $ pst -un
   Processor | PRIORITY | Program | State | UNIX INFORMATION | Process
   Time (sec) | mn/cu/mx | Counter | | PID | PPID | PGID | Name
   _____
    1016.261 16/16/16 18088 Wait 1 0
                                                       1 display_manager
    7.269 1/2/16 17C30 Wait 23 1 23 netman
     4.385 1/15/16 17C30 Wait 24 1 24 mbx_helper
10.939 1/15/16 17F78 Wait 26 1 26 server_process_manager
   -----
     1038.855
4. $ pst -pa
Processor | PRIORITY | Program | State | Private | Global | D I S K | N E T | Process
Time (sec) | mn/cu/mx | Counter | | Faults | Faults | Page IO | Page IO | Name
288.549 16/16/16 18088 Wait 2168 783 0 4180 display_manager 444.001 1/15/16 <active> Ready 6284 1557 0 11097 process_1 72.018 1/16/16 17E90 Wait 124 26 0 151 digital_clock 7.793 1/15/16 17E90 Wait 279 120 0 436 alarmer 0.715 1/15/16 17C30 Wait 36 6 0 84 mbx_helper 1.502 1/16/16 17F78 Wait 104 51 0 175 mail 0.715 1/16/16 17F78 Wait 44 22 0 74 shell
-----
                                    [----|---|
```

9039 2565 0 16197

815.296

RBAK (READ_BACKUP) -- Restore or index a magnetic media backup file.

FORMAT

RBAK {pathname [-AS disk_pathname] ... | -ALL} [options]

RBAK restores a magnetic media backup file which was written with WBAK (WRITE_BACKUP). Use WBAK and RBAK to backup disks and to transfer information between separate DOMAIN installations. (Use the RWMT (READ_WRITE_MAGTAPE) command to transfer information to and from non-DOMAIN installations.)

RBAK operates in either "index" or "interchange" mode. To restore objects to disk, use interchange mode (-INT). To list object names on standard output, without restoring any information to disk, use index mode (-INDEX).

When using RBAK, please note the following:

- RBAK must be run on the node which is connected to the tape or floppy drive unit. You may accomplish this either by physically typing the RBAK command on the host node, or by running RBAK in a process on the host node created from your own remote node using the CRP CREATE_PROCESS command.
- There is no special tape mounting command. Simply mount the tape on the tape drive and execute RBAK.
- Only one tape unit can be connected to any node.
- The directories on disk must have delete access to be replaced by directories from the backup media. The disk directories must have append access for files to be added from the backup media.
- Files on disk must have delete access to be replaced by files from the backup media.
- Locked objects cannot be restored from backup media. See the LKOB (LOCK OBJECT) command.

ARGUMENTS

pathname (optional)

Specify name of object to be indexed or restored to disk. This may be a directory, file, or link. If the object is being restored, the new disk object will have the same name. If you wish the disk file to be saved under a different name, use -AS (below). Multiple pathnames are permitted; however, wildcarding is not supported.

Default if omitted: must use -ALL.

OPTIONS

Default options are indicated by "(D)."

Backup File Identifiers

One of the following options is required.

-F file no

Read the backup file with the file number specified. You

assigned this number with WBAK.

-F CUR

Begin reading at current position on the backup media.

-FID file id

Read the backup file name specified. You assigned this name

using WBAK.

Mode Control

-INT

(D)

Select "interchange" mode. Backup files are restored to disk.

-INDEX

Select "index" mode. Backup file names are listed on standard

output; no information is restored to disk.

Catalog Control

-ALL

Restore or index all the objects in the backup file specified. This option is required if you do not use the 'pathname' argument to indicate a particular object to be indexed or restored.

-AS pathname1

Restore the object specified and assign a different disk pathname ('pathname1'). This option is only valid when used with the

'pathname' argument on the RBAK command line.

-CR (D)

-FORCE

Specify create mode. RBAK does not restore objects if their names already exist on disk. It prints an error message if a

name exists on both disk and backup media, and continues.

Specify replace mode. RBAK deletes the existing disk object, and replaces it with the object read from backup media.

Force object deletion if you have owner rights, even if you don't

have delete rights.

-DU Delete when unlocked. If the object to be deleted is locked when

RBAK is invoked, the delete operation will be performed when

the object is unlocked.

-MS Specify merge-source mode. Similar to replace mode. If an

object already exists on disk, RBAK deletes the disk version and restores the backup media version (the source). However, if the object is a directory, RBAK merges the backup media directory's

contents with the disk directory.

-PR pathname...

Preserve specified objects on the disk. Multiple pathnames and wildcarding are permitted. If the objects exist on disk, they will NOT be overwritten by backup media versions. This option must be used with the -MS option.

-MD

Specify merge-destination mode. Similar to create mode. If an object already exists on disk (the destination) RBAK does not restore the backup media version, and retains the disk version. However, if the object is a directory RBAK merges the backup media directory's contents with the disk directory.

Label Control

-SLA (D) Display the backup media file label on standard output.

-NSLA Do not display the backup media file label.

Listing Control

You may include the -L option, or any combination of -LD, -LF and -LL.

-L Write all the file, directory, and link names to standard output.

-LD Write all directory names to standard output.

-LF Write all filenames to standard output.

-LL Write all linknames to standard output.

Backup Device Control

-ANYS

Force RBAK to accept any section of the backup file. When a backup file spans multiple backup media volumes, RBAK normally begins with the backup media volume containing the backup file's first section, and proceeds to the backup media volume containing the second section, and so on. If you know which backup media volume contains the object you want to restore or index, use this option. This lets RBAK start at any section of the backup file.

-REO

Force previous volume to be reopened, and suppress reading of backup media volume label. Use only when backup media has not been repositioned since last WBAK or RBAK.

-DEV d[unit]

Specify device type and unit number. 'd' must be either 'M' (for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., -DEV M2). A unit number is NOT required for floppy disks and cartridge tapes (i.e., -DEV F). If this option is omitted, RBAK assumes device M0.

CAUTION: Floppy disk support for this command is limited.

In particular, error detection during reads and

writes is poor. DO NOT use this command with floppy disks when the data being placed on the floppy disks are critical and unrecoverable.

-RETEN

Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.

-NRETEN (D)

Do not retension the cartridge tape.

-REWIND

Rewind the cartridge tape after reading or indexing. If this option is omitted, the cartridge tape is left positioned to the next tape file. This option is valid ONLY for the cartridge tape; reel-to-reel tapes get rewound automatically when removed from the drive.

ACL Control

-DACL

(D) Assign the destination directory's default ACL to the object

being restored.

-SACL

Retain the restored object's orginal ACL.

DTM/DTU Control

-PDT

Preserve the object's original date-time modified and date-time

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ rbak -f 1 fred/soup Read "fred/soup" in backup file 1 and restore it
to disk. "Fred/soup" may be a directory, file,
or link.

\$ rbak -f 1 fred/soup -as //node5/noodle Restore "fred/soup" and place
it in "noodle" on node5.

\$ rbak -dev ct -rewind Rewind the cartridge tape prior to removing it from the tape unit.

DIAGNOSTICS

I/O Errors

When RBAK encounters an I/O error, it attempts the operation again, for a total of five times. After the fifth attempt fails, RBAK prints out an error message describing which type of error occurred. If the error happened during an attempt to read from a tape, RBAK skips the tape block which it could not read, and tries to read the next one. (Note that a tape block is 8 Kb

long.) If it again fails, after five tries, it skips that block and tries the next. This process will continue for a total of twenty consecutive failed blocks, at which time RBAK aborts.

tape rewind error

An I/O error occurred.

tape write-filemark error

An I/O error occurred.

tape space-filemark error

An I/O error occurred.

tape space-record error

An I/O error occurred.

i/o recovery failed

An I/O error occurred and the tape drive could not reposition for another try.

tape i/o error

An I/O error occurred.

tape i/o error -- data lost

An unrecovered I/O error occurred and data was lost.

Operator Errors

first label on volume is not VOL1 label

Expected a standard label, and did not find one. The tape was not written with WBAK.

label version number in VOL1 label is not "3"

The label format is incorrect. The tape was not written with WBAK.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This error can occur only when a multi-tape file is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

file not found

The tape file specified was not found.

invalid unit number

Tape unit specified is not connected. Presently, only DEV M0 is supported.

pbu is not present.

No tape unit is connected to the node. RBAK can only be run on the node connected to the tape drive.

RDYM (READY_MESSAGE) -- Set system ready message.

FORMAT

RDYM (-ON | -OFF)

RDYM enables or inhibits the output of a system ready message to standard output after execution of each Shell command. The message lists the CPU time required to execute the command and the elapsed time since the last command. Both times are reported in seconds and decimal fractions of seconds. The message appears on a line following the echoed command line in the Shell's process transcript pad.

Turning on the ready message interactively or in a Shell script causes it to be printed after each command of the program is executed. If the ready message is not disabled at the end of the Shell script, it will remain in effect after the Shell script exits.

Ready message printing is enabled and disabled for *levels*. The level number increases each time a Shell script or the Shell is invoked, and decreases when it exits. The times printed in the ready message reflect the CPU and real time used since the last message at the same level. Thus, for example, if the ready message is enabled in a Shell script, after the last command of the program is finished, two ready messages will be printed: one showing the time used by that command, and one showing the time used by the whole Shell script.

If the ready message is turned on by one level, it will remain on when that level exits; however, if it is disabled by a level, it will revert to its previous state when that level exits.

By default, system ready messages are disabled at login.

OPTIONS

-ON

Enable message.

-OFF

Disable message.

EXAMPLES

```
$ rdym -on
cpu time: 750.685. real time: 5914.532.
$ bldt
AEGIS, revision 6.0, built on Friday, April 15, 1983 9:26:30 am (EST).
cpu time: 0.234. real time: 1.736.
$ rdym -off
$
```

READ -- Set variables equal to input values.

FORMAT

READ [options] {-TYPE type var_name ... | variable_list}

The READ command reads input values and sets a list of variables to those values. The values from the input line are parsed as seperate tokens (they must be seperated by spaces), and each variable in the list is assigned the value of a token.

Use the "-P prompt" argument to instruct READ to issue a prompt. If you do not input values for all the variables names listed as part of the READ command, READ displays a "<more>" prompt to request further input.

By default, the type of each variable specified in the READ command depends on the type of each input value. You can, however, use the -TYPE argument to specify the individual type(s) of the the variables.

ARGUMENTS

variable_list
(optional)

Specify the names of the variables that receive the input values.

Default if omitted: must specify -TYPE (below).

OPTIONS

-TYPE type var name...

Specify the type of the input value(s) that can be assigned to the particular variable name(s). Multiple variable names are permitted, separated by blanks. Once you specify a type in a particular READ command, READ assigns that type to all subsequent variable names, until you change the type specification. Valid types are

STR[ING] character strings
INT[EGER] integer numbers
BOOL[EAN] Boolean values
ENV[IRONMENT] environment variables
ANY any type (the default)

If the type of the input value does not match the type specified for that variable name, READ issues an error and asks you enter another input value. Use -TYPE ANY to restore the Shell to its default state. In this case, it determines the proper variable type automatically.

Specifying -TYPE ENV var_name causes the variable to become an environment variable. Environment variables are of primary concern to DOMAIN/IX users; please consult the DOMAIN/IX documentation for details about their usage.

-P[ROMPT] prompt

Specify a particular prompt string to request the input values. Enclose the string in single quotes if it contains literal blanks.

-ERR[IN]

Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READ command appears inside a pipeline, since standard input in that case is connected to the pipe.

EXAMPLES

Consider the following command line in a Shell script:

READ -P "Enter model and class:" model class

In this example, READ displays the prompt "Enter model and class:" in the process input window, and assigns the input values to the variables named "model" and "class", in that order.

The following section illustrates how the -TYPE option works. (The numbers in parentheses are used to refer to the different parts of the example.)

```
$ READ -P '> ' -TYPE integer tens ones -TYPE string number (1)
> 40 four
<non-integer 'four'; please reenter> > 4 (2)
<more> > forty-four (3)
$
$ LVAR
integer tens = 40
integer ones = 4
string number = forty-four
$
```

In line (1) we define the prompt to be "> ", specify variables "tens" and "ones" of type "integer", and specify variable "number" of type "string". This means that the READ command expects its input to be three variables of types integer, integer, and string, in that order. When we enter the non-integer value "four", READ cannot assign this value to variable "ones", and issues the error message and prompt shown in line (2). In line (3) READ prompts for the third input value. The LVAR command, issued in line (4), displays the type, name, and value of the variables.

Here is a final example.

```
$ date | chpat ',' '' | (read day month date year; readIn time)
$ lvar
string time = 12:40:42 pm (EST)
integer year = 1985
string month = January
string day = Wednesday
integer date = 2
$
```

In this example, the output from the DATE command is piped to CHPAT, which removes the commas and then sends its output to READ and READLN where the proper variable assignments are made.

READC -- Set variables equal to input characters.

FORMAT

READC [options] variable_list

The READC command reads single characters as input, and sets a list of variables equal to those character values. READC parses each character from the input line as a separate token, and each variable in the list is assigned the value of a token. Use the "-P cprompt>" argument to instruct READC to issue a prompt.

The READC command considers all input to be type "string".

ARGUMENTS

variable_list

(required)

Specify the names of the variables that receive the input values.

OPTIONS

-P[ROMPT] prompt

Specify a particular prompt string to request the input values. Enclose the string in single quotes if it contains literal blanks.

-ERR[IN]

Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READC command appears inside a pipeline, since standard input in that case is connected to the pipe.

EXAMPLES

Consider the following sequence of commands and input:

```
$ readc -p "Do you want to continue? (y/n): " ans
Do you want to continue? (y/n): y
$ lvar
string ans = y
```

In this example, READC displays the prompt "Do you want to continue? (y/n): " in the process input window, and assigns the value of the first input character ("y" in this case) to the variable named "ans".

For more information on Shell variables, refer to the DOMAIN System User's Guide.

READLN -- Set a variable equal to an input value

FORMAT

READLN [options] variable_list

The READLN command reads a line of input and sets a variable to that value. Use the "-P rompt> " argument to instruct READLN to issue a prompt. READLN accepts
multiple variable names.

The variable type is always a string.

Refer to the descriptions of the READ and READC commands for related information.

ARGUMENTS

variable_list

(required)

Specify the name(s) of the variable(s) that receives the input value(s). If you specify more than one variable name (separated by blanks), READLN assigns the values of input lines to the variables in the order that the variables were named.

OPTIONS

-P[ROMPT] prompt

Specify a particular prompt string to request the input value. Enclose the string in single quotes if it contains literal blanks.

-ERR[IN]

Read input from error input instead of standard input. This option is useful for reading user input from the Shell's input pad (where error input is normally directed) when the READLN command appears inside a pipeline, since standard input in that case is connected to the pipe.

EXAMPLES

Consider the following command line in a Shell script:

READLN -p "Enter total here: " total

In this example, READLN displays the prompt "Enter total here: " in the process input window, and assigns the value of the input line to the variable named "total."

RETURN -- Return from current Shell level.

FORMAT

RETURN [options]

The RETURN command causes the Shell to return from its current level with the specified status severity. See the ABTSEV command description for details about status severity levels.

OPTIONS

Specify one of the following options to select the return severity level. All options must be specified in UPPERCASE letters.

Default options are indicated by "(D)."

-OK

Set level to OK.

-T[RUE]

(D)

Set level to true.

-F[ALSE]

Set level to false.

-W[ARNING]

Set level to warning.

-E[RROR]

Set level to error.

-O[UTINV]

Set level to output invalid.

-I[NTFATAL]

Set level to internal fatal error.

-P[GMFLT]

Set level to program fatal error.

-M[AX_SEVERITY]

Set level to maximum severity error.

EXAMPLES

The following lines are a portion of a Shell script:

```
# Test to see if the second parameter passed to the script is valid.
# If it is not, abort the script.
if eqs ^1 '-test' then
   if eqs ^2 '-b' then
      cpf ^3 temp.mss
   else
      args "(?) >>> ^2 <<< is not a valid parameter."
      return -P
   endif
else
   cpf ^1 temp.mss
endif</pre>
```

REVL (REVERSE_LINES) -- Reverse each line in a file.

FORMAT

REVL [pathname ...]

REVL copies the named files to standard output, reversing the order of the characters in every line.

ARGUMENTS

pathname

(optional)

Specify name of file containing lines to be reversed.

Default if omitted: read standard input.

EXAMPLES

1. \$ rev1
 This command produces interesting results.
 .stluser gnitseretni secudorp dnammoc sihT
 *** EOF ***
\$

Reverse a line from standard input.

2. \$ revl /sys/dict | srf | revl >rhyming_dict
\$

Sort the system dictionary by suffixes to produce a rhyming dictionary. RWMT (READ_WRITE_MAGTAPE) -- Read/write foreign magtapes.

FORMAT

RWMT mode _ control [pathname...] [options]

RWMT reads tapes from non-DOMAIN installations and writes tapes which can be read by non-DOMAIN installations. RWMT can read and write unlabeled tapes, as well as ANSI level 1-4 labeled tapes.

For information on reading and writing tapes intended for exchange with other DOMAIN installations, see the RBAK (READ_BACKUP) and WBAK (WRITE_BACKUP) commands.

When using RWMT, please be aware of the following:

- RWMT must be run on the node which is connected to the tape unit. You may accomplish this either by physically typing the RWMT command on the host node, or by running RWMT in a process on the host node created from your own remote node using the CRP CREATE PROCESS command.
- Only one tape unit can be connected to any node.

ARGUMENTS

pathname

(optional)

Specify name of file to be read from or written to tape. This argument is only valid with the -R and -W mode control options (below). Multiple pathnames and wildcarding are permitted.

Default if omitted: read pathnames from standard input.

OPTIONS

Default options are indicated by "(D)."

Mode control

One of the following mode control options must be specified. If you omit it, RWMT will prompt you for it. You may have RWMT prompt for all necessary options by using the -P option.

-L[ABEL]

Write ANSI X3.27-1978 volume label on a tape. This option causes RWMT to write an ANSI volume label and dummy file on the magtape volume. An optional owner and volume ID, which are stored in the volume label, may be specified (see -VID and -OWN below). This is the way to initialize a labeled tape; if any information existed on the tape, it is erased by this labeling operation.

If you are labeling a tape, then the following two options may also be used.

-VID vol_id Specify a 1-6 character volume ID for use when labeling a volume. This option is only valid when used with the -L mode_control option (above). The default volume ID is '' (blank).

-OWN owner id

Specify a 1-14 character owner ID for use when labeling a volume. This option is only valid when used with the -L mode control option (above). The default owner ID is '' (blank).

-I[NDEX]

List objects on an ANSI-labeled physical tape volume. -INDEX produces a listing of all files or file sections on an ANSI-labeled physical tape volume. The contents of the physical volume (VOL1) label and all file header labels are written to standard output.

-W[RITE]

Specify one or more disk files ('pathname' argument) to be written to tape. The default format is ANSI labeled, ASCII, fixed-length records of 80 bytes each, and 80-byte blocks. If desired, any of these parameters can be changed using the options described below. If more than one pathname is specified, the disk files are written to sequential tape files. Tapes written by RWMT are always in accordance with ANSI level 4 format.

Before writing a labeled file, the tape volume itself must be labeled with the -LABEL mode control option (above).

-R[EAD]

Specify one or more tape files to be read from tape and stored on disk. READ reads one or more tape files and writes them to disk using the specified pathnames ('pathname' argument). The default tape file format is the same as that for the WRITE option. If the tape is labeled under ANSI level 2, 3, or 4, the file format (block length, record length, and record format) is read from the tape. If the tape is unlabeled, or labeled with ANSI level 1, you must specify the tape format using the options below. If more than one pathname is specified, adjacent tape files are read and stored under the specified pathnames.

Label Control

-ANSI (D) Specify that the tape is labeled in conformance to ANSI X3.27-1978, level 1, 2, 3, or 4.

-UNLAB

Specify that the tape is unlabeled.

Tape Format

-ASC (D) Specify that all tape file contents are in ASCII characters.

-EBC

Specify that all tape file contents (except labels) are in EBCDIC characters.

-RAW

Specify that all tape file data is to be treated in raw form (see example 5).

-NPAR

(D)

Specify no disturbance of parity bits when reading or writing data.

-PAR

Specify that parity bits should be forced off when reading data from tape and forced on when writing data to tape.

-RL reclen

Specify the maximum length, in bytes, of a record in the tape file. This option is only valid when used with either the -R or -W mode control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default record length is 80 bytes.

-BL blocklen

Specify the length, in bytes, of a physical tape block. This option is only valid when used with either the -R or -W mode control options (above). It is unnecessary when reading an ANSI level 2, 3, or 4 file. The default block length is 80 bytes.

-BF blockfac

Specify a blocking factor -- the number of records to store into or read from a physical tape block. This is an alternative to the -BL option, since the record length multiplied by the blocking factor yields the block length. This option is only valid when used with either the -R or -W mode control options (above). Using this option is only meaningful if your tape has fixed-length records. This option is unnecessary when reading an ANSI level 2, 3, or 4 file. The default blocking factor is 1.

-RF format

Specify record format. Valid values for 'format' are "F" (fixed-length records and blocks); "D" (variable-length records (this is ANSI 'D' format)); "S" (spanned records); or "U" (undefined record format). The default format is "F." Note that if you are writing a cartridge tape, only 512 byte blocks may be written; D, S, and U formats are not supported.

Tape File Identifiers

-FID file id

Specify a 1-17 character file ID to be written in the file header label for use when writing a file to a labeled volume. This option is only valid when used with the -W mode control option (above). If this option is omitted, the name of the file being written is used.

-F [position]

Specify the file position for -R or -W operations. Valid values for 'position' are "CUR," "END," or a nonzero integer value. A position of "CUR" selects the current tape position; the tape must have been previously read or written by RWMT and its position must not have been disturbed. This option is only valid when used with either the -R or -W mode control options (above).

A position of "END" selects the end of the tape file set. This option is valid only when used with the -W mode control option, and causes RWMT to append the specified disk file ('pathname' argument) to the very end of the tape file set.

RWMT (READ_WRITE_MAGTAPE)

A position specified by a nonzero integer value selects the file at that absolute position in the tape volume. This option is only valid when used with either the -R or -W mode control options (above). If multiple 'pathname' arguments are supplied, the value of 'position' is incremented by one after each file has been read or written.

The default value for 'position' is 1.

Backup Device Control

-DEV d[unit]

Specify device type and unit number. 'd' must be either 'M' (for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., -DEV M2). A unit number is NOT required for floppy disks and cartridge tapes (i.e., -DEV F). If this option is omitted, RBAK assumes device M0.

-RETEN

Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.

-NRETEN (D)

Do not retension the cartridge tape.

Miscellaneous Control Options

-SBIN

Cause all stream files written to contain the binary attribute (normally, output stream files contain the ASCII attribute).

-P

Cause RWMT to prompt for all unspecified parameters.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

- 1. \$ rwmt -label -own "R and D" -vid "demo" Initialize a tape with the given owner and volume ID.
- 2. \$ rwmt -w c?*_example -f 1 -rf d -rl 200 -bl 2048 Copy the
 32 records of "cmf_example" written to tape file 1. wildcarded
 8 records of "cmt_example" written to tape file 2. files to
 4 records of "cpboot_example" written to tape file 3. tape.
 25 records of "cpf_example" written to tape file 4.

3. \$ rwmt -index

List the files on the tape.

Access: " "

Volume label:

Volume ID: "DEMO "

File/S	ection	File ID	Cr Date	Acc RF	RL	BL
1	1	CMF EXAMPLE	83/02/17	D	200	2048
2	1	CMT EXAMPLE	83/02/17	D	200	2048
3	1	CPBOOT_EXAMPLE	83/02/17	D	200	2048
4	1	CPF_EXAMPLE	83/02/17	D	200	2048
5	1	CPT EXAMPLE	83/02/17	D	200	2048

Owner ID: "R AND D

End of file set.

\$

4. \$ rwmt -r cpboot_example.tape -f 3
4 records read from tape file 3 into
 "cpboot_example.tape".

Copy tape file 3 to a disk file named "cpboot_example.tape"

5. RAW mode

RWMT permits a tape file to be read in RAW mode. In this mode, each block read from the tape is written into one record in a stream file, unmodified by the program. To read a file in RAW mode, you should specify the maximum record size using the -RL argument. If you do not, the default value of 80 bytes is used, and any records longer than that are truncated. Also, undefined record format should be used. For example:

\$ rwmt -r -f 1 -rf u -raw -rl 512 rawfile

reads tape file number 1 into "rawfile", with a maximum record length of 512 bytes.

Files may be written in the same manner:

\$ rwmt -w -f 1 -rf u -raw -rl 512 rawfile

DIAGNOSTICS

I/O Errors

When RWMT encounters an I/O error, it attempts the operation again, for a total of five times. After the fourth retry fails, RWMT prints out an error message describing which type of error occurred. If the error was during an attempt to read from a tape, RWMT skips the tape block which it could not read, and tries to read the next one. If it again fails, after five tries, it skips that block and tries the next. This process may continue for a total of 20 consecutive failed blocks, at which time RWMT aborts. If the error occurred while writing a block to tape, RWMT aborts after the fifth attempt to write to that block.

tape rewind error

An I/O error occurred. An illegal operation may have been attempted, such as trying to perform an INDEX on an unlabeled volume.

i/o recovery failed

An I/O error occurred and RWMT could not position the tape for another try due to a separate I/O error.

RWMT (READ_WRITE_MAGTAPE)

tape write-filemark error An I/O error occurred.

tape space-filemark error An I/O error occurred.

tape space-record error
An I/O error occurred.

tape i/o error
An I/O error occurred.

Format Errors

The following errors indicate that the labels of a labeled tape do not conform to ANSI standards.

first label on volume is not VOL1 label Expected a standard label, and did not find one.

label version number in VOL1 label is not "3"

The label format is incorrect.

a HDR1 label is missing where one is required

A file on the tape does not begin with the correct format.

an EOF1 (or EOV1) label is missing where one is required

A file on the tape does not end with the correct format.

a double filemark was encountered unexpectedly

The tape format is incorrect.

inconsistent file sequence numbers

The tape format is incorrect.

file sequence number tracking error The tape format is incorrect.

variable record with invalid record control word encountered.

The tape format is incorrect.

spanned record with invalid segment control word encountered The tape format is incorrect.

erroneous spanning indicator or segment out of sequence encountered The tape format is incorrect.

Operator Errors

The following errors indicate that the equipment was not set up correctly or, that the tape was not loaded correctly.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This error can occur only when a multitape volume is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

Miscellaneous Errors

file not found

The tape file specified is not in the volume.

invalid record format specifier

The record format specified must be D, F, U, or S.

block size is too large

Block size specified is larger than 16 Kb.

conflicting blocking information

Record size specified is larger than block size specified for "F" record format.

invalid unit number

Tape unit specified is not connected. Unit numbers must be between zero and three inclusive.

pbu is not present

No tape unit is connected to the node. RWMT can only be run on the node connected to the tape drive.

This file contains n invalid records which were not written to tape

N records of length less than 4 were in the original disk file. Such records cannot be written to tape. This message is just a warning; the rest of the file was written to the tape.

SALACL (SALVAGE_ACLS) -- Salvage an Access Control List.

FORMAT

SALACL [options] [volume]

SALACL salvages the ACL structure on the volume you specify. It corrects the reference counts on all ACL objects. Reference counts record the number of files that use each ACL. After it repairs reference counts, SALACL merges duplicate ACLs into a single copy, and deletes unused ACLs. You should run SALACL once a week or so unless you always receive reports that no reference counts needed repairing (i.e., everything is perfect).

SALACL cannot merge duplicate ACLs on files that are currently in use (locked) (for example, library files). This is not especially serious, as the duplication consumes very little disk space. To merge all possible ACLs on a node, including things like libraries, bring the node up diskless, mount the volume using MTVOL, and then run SALACL on the mounted volume.

ARGUMENTS

volume

(optional)

Specify the entry directory pathname for the volume whose ACLs you intend to salvage. Note that SALACL cannot salvage

a volume mounted on a remote node.

Default if omitted: '/' (node entry directory)

OPTIONS

-N[O_]S[UMMARY]

Suppress summary information.

-V[ERIFY]

Verify only; do not delete or merge any ACLs.

-N[O-]M[ERGE]

Do not merge duplicate ACLs into a single copy.

EXAMPLES

\$ salacl

Salvage the ACLs of the current volume.

Warning: unable to merge two equivalent ACLs:

//grover/sys/node_data/Shell - object is in use (OS/file server)

ACL objects found:

ACL reference counts fixed:

ACLs garbage collected:

ACLs in use during g.c.:

ACLs merged with equivalents:

12

SALD (SALVAGE_DIRECTORY) -- Salvage a directory.

FORMAT

SALD pathname...

SALD corrects problems in directories caused by a system crash or network failure. The specified pathname must refer to a directory.

SALD makes the specified directory usable and saves as much information as possible.

The following are symptoms of damaged directories:

- LD reports that the directory is empty, but it cannot be deleted.
- LD lists a file in the directory, but no other command can find the file.
- The directory is completely unreadable, and errors occur on every access attempt.

ARGUMENTS

pathname

(required)

Specify name of directory to be salvaged. Multiple pathnames and wildcarding are permitted.

OPTIONS

SALD has no unique options.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ sald /sqh/data_dir

Salvage the directory specified.

SALRGY (SALVAGE_REGISTRY) -- Salvage (update) the network registry.

FORMAT

SALRGY [options]

SALRGY maps the registry data files from all the sites in the network registry, determines the latest one and copies that version to all the sites. You will probably not be able to use this command unless you are the network administrator for your network, since SALRGY requires you to have write access to the network registry database. For complete information on the use of the network registry, see Administering Your DOMAIN System.

OPTIONS

-R pathname

Specify the node entry directory name of a node containing a registry file copy. If you omit this option, SALRGY obtains the registry pathnames from the current node's registry file copy (/REGISTRY/REGISTRY).

EXAMPLES

\$ salrgy -r //os

Salvage the network registry using the registry file copy on node "os".

SALVOL (SALVAGE_VOLUME) -- Verify and correct allocation of disk blocks.

FORMAT

From AEGIS: SALVOL

From Mnemonic Debugger: EX SALVOL

Each logical volume is divided into disk blocks. SALVOL verifies and, if necessary, corrects the tables that describe the allocation of disk blocks to the files stored on the disk. SALVOL also returns to the free space pool all blocks that are no longer in use: those allocated to temporary files or to deleted portions of permanent files.

SALVOL can usually restore a disk after a system crash or an improper dismount of a volume. Please refer to the appropriate appendix for complete information on the use of this software tool.

SALVOL prompts for all required arguments and options.

SCRTO (SCREEN_TIMEOUT) -- Set/show screen timeout

FORMAT

SCRTO [n] [-NONE]

The SCRTO command sets or displays the number of minutes after the last input event that the Display Manager waits before it shuts off the display screen.

By default, the DM waits 15 minutes before it shuts off the display.

ARGUMENTS

n

(optional)

Set the number of minutes for the DM to wait before it shuts

off the display.

Default if omitted: display current timeout setting.

OPTIONS

-NONE

Disable automatic timeout; never turn off the display.

EXAMPLES

\$ scrto

The screen timeout is set to 15 minutes

\$ scrto 10

\$ scrto

The screen timeout is set to 10 minutes

\$ scrto -none

\$ scrto

Screen timeout is currently turned off

\$ scrto 15

\$

Show initial setting.

Set delay to 10 min.

Disable timeout.

Restore intial setting.

SELECT -- Execute a SELECT statement.

FORMAT

```
SELECT arg [mode]

CASE arg [TO arg]

[command...]

[CASE...

command...]

[OTHERWISE

command...]

ENDSELECT
```

SELECT allows you to build a control structure that executes commands according to the results of one or more Boolean tests. The Shell uses each CASE clause to perform a separate Boolean test on the initial SELECT argument. If the CASE argument is equal to the SELECT argument, the result of the test is TRUE and the command(s) within the CASE clause execute.

You may test multiple CASEs simultaneously. If you place several CASE clauses on the same line (or specify line continuation with @<RETURN>), the CASEs are logically OR'd and return TRUE if any one of the CASEs is TRUE. (See example below.)

The (optional) TO clause allows you to specify an integer or string range for testing. For example, you might specify "CASE 0 TO 9" to test for any single digit, or "CASE a TO z" to test for a lowercase letter.

The (optional) OTHERWISE clause executes if and only if none of the CASE clauses returns TRUE (regardless of whether the selection mode was 'ONEOF' or 'ALLOF').

ARGUMENTS

arg

(required)

Any valid token, defined integer or string variable, or expression. SELECT compares the first 'arg' with each of the CASE 'arg's to determine which command(s) to execute.

mode

(optional)

Specify selection mode. Valid modes are 'ONEOF' and 'ALLOF'. If you specify ONEOF (the default), SELECT executes only the first CASE statement that returns a TRUE value. If you specify 'ALLOF', SELECT executes all CASE statements that return TRUE.

Default if omitted: use 'ONEOF'.

command...

(optional)

Specify the command to be executed when the CASE test returns TRUE. This may be a Shell command, a Shell script, a variable assignment, or any other valid Shell operation. Multiple commands are permitted; separate them with semicolons or NEWLINE characters.

Default if omitted: no command executed for this CASE clause.

EXAMPLES

```
select ^a allof
   case 1 case 2 case 3
       args "This will print if a = 1 or a = 2 or a = 3"
   case 1 @
   case 2 @
   case 3
       args "This is the same test as the previous one, since the"
       args "carriage returns are escaped."
   case 4 # This is a case without a body to execute.
   case 5 to 10
       args "This will print if ^a is in the range 5-10."
   case 6
       args "This will also print if ^a = 6, since ALLOF is"
       args "specified."
  otherwise
       args "This will print if \hat{a} is not between 1 and 10."
{\tt endselect}
```

SEND_ALARM -- Send messages to alarm servers.

FORMAT

SEND_ALARM string ... {[target_option ...]} [-L]

SEND_ALARM transmits one-line messages that ALARM_SERVER can display. You can direct messages to

- a particular user, specified by the name used for login,
- the user on a particular node, specified by the node's entry directory or node ID,
- the users on all of the diskless nodes that have a particular paging partner,
- or combinations of these categories of users.

You must always specify at least one user to receive the message, but you may use any of these techniques for choosing users.

Messages are not stored and message delivery is not guaranteed. If the intended recipient is not running an alarm server when the message is sent, or if other problems arise, the message is never delivered. SEND_ALARM notifies you when messages can not be delivered.

ARGUMENTS

string ... (required)

Specify the message you want to send. Multiple strings are permitted, separated by blanks. If several strings are present, they are concatenated with intervening spaces to form the message.

OPTIONS

-L List the target nodes or users as the messages are sent.

At least one of the following target options must be specified.

-U[SER] login_name ...

Specify one or more users to whom the message should be sent. If you use this option more than once, the several lists of users are concatentated. Every user on the concatenated list receives the message.

If a user is logged on several nodes at the same time, and has alarm servers running at each of those login sessions, only one of the alarm servers will be able to receive the messages. -N[ODE] node_spec ...

Transmit the message to whatever user is logged in on the specified node, if any. 'node_spec' may be an entry directory (e.g. //GRUNGE), or a hexadecimal node ID. Multiple node spec's are permitted. If you use this option more than once, the several lists of nodes are concatentated. Every user on a node on the concatenated list receives the message.

If several users are logged in on one node simultaneously and are running alarm servers, only one user is able to receive messages directed to that node.

-DI[SKLESS] node_spec ...

Transmit the message to whatever user is logged in on any diskless node whose host node is specified by 'node_spec', as in the -NODE option. Multiple node spec's are permitted. If you use this option more than once, the several lists of nodes are concatentated. Note that this option does not send a message to the user on the paging partner. See -DIN below.

-DIN node_spec ...

This option allows you to send a message to a disked node and all of its diskless partners. This is the same as specifying "-NODE node_spec ... -DISKLESS node_spec ... ".

-ME or -MYU[SER]

Adds you to the list of users to receive the messages.

-MYN[ODE]

Adds the node at which the command is entered to the list of target nodes.

-MYDI[SKLESS]

Same as "-DISKLESS node_spec", where 'node_spec' is the node at which the SEND_ALARM command was entered.

-MYDIN

Same as "-MYNODE -MYDISKLESS".

-ALLN[ODE]

Send the message to every node seen by an LCNODE.

EXAMPLES

- \$ SEND_ALARM 'Meeting is cancelled' -USER joe_k sue_b john_f mary_g
- \$ SEND_ALARM Please log out. Node AAD going down. -DIN Oaad
- \$ SEND_ALARM Compilation completed -ME
- \$ SEND_ALARM Power going off at 17:30 tonight -ALLNODE -L

\$ SEND_ALARM 'Loop 13 is being switched out' *loop13

Note: In this example, "loop13" is a file in the working directory containing further input for the SEND_ALARM command line. It might, for example, contain the text:

-N //dirtball //thorazine

-N //top 317F

-U network manager

-MYDIN

SET -- Set current Shell conditions.

FORMAT

SET [options]

SET allows you to change the state of the current Shell. It accepts the same options as the SH (SHELL) command, thus permitting you to alter current Shell conditions without having to generate an entirely new Shell. See the SH command description for general information about Shells.

Separate Shell commands already exist for altering a few of the current Shell conditions (EON | EOFF; VON | VOFF; etc.). SET combines all of those functions into a single facility.

OPTIONS

Default options are indicated by "(D)."

-B[ON]

Send the output of a background process (created with the & parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it was invoked. If you do not specify -B, the output of the background process is sent to /DEV/NULL.

- -BOFF
- **(D)** Do
- Do not display output from a background process.
- -NB[ON]
- (D) Same as -BOFF.
- -C[OMMAND] arg1 ...

Execute the following argument(s) as a Shell command, exactly as if it had been read as an input line. If any argument contains explicit blanks, enclose it in quotes. The Shell passes all text following -C to 'arg1' as arguments, so if you want to specify other options to the SH command itself, they must precede -C.

-E[ON]

Enable evaluation of variables outside of expressions. If -E is specified, the Shell always evaluates variables, regardless of the context in which they appear. If -E is not specified, variables are evaluated only inside variable expression delimeters, ((expression)); otherwise, the Shell treats the ^var_name expressions as strings and they are not evaluated.

- -EOFF
- (D)
 - Evaluate variables only inside expressions.
- -NE[ON]
- (D) Same as -EOFF.
- -I[NTER]

Behave as though input is being entered interactively: prompt for each input line, and do not exit on errors or quit faults (DQ or CTRL/Q from keyboard). Normally, the Shell only executes interactively if its input comes from a pad or sio-line. Use of this option forces prompting.

-S[CRIPT] (D) Behave as though executing a Shell script: do not prompt and abort on error. A Shell normally will not quit; any error or quit command is assumed to apply only to the last command given to the Shell.

-NI[NTER] (D) Same as -S.

-N[EXECUTE]

Interpret each command line only; suppress execution.

-EX[ECUTE] (D) Interpret each command line and execute it.

-P[ROMPT]1 prompt_string

Define the prompt string for the Shell created with SH.

-P[ROMPT]2 subprompt_string

Define the subpromt string for the Shell created with SH. (The subprompt appears when you continue a Shell command over more than one line).

-START [file] (D) Execute the specified script after the Shell is created. If 'pathname' is not specified, the Shell searches for a file called ~USER_DATA/STARTUP_SH.INIT and executes it if it exists. No error occurs if that file does not exist.

-NSTART Disable startup file execution.

-V[ON] Display each line of text in the transcript pad as it is read by the Shell program.

-VOFF (D) Disable input verification.

-NV[ON] (D) Same as -VOFF.

-X[ON] Display each command in the transcript pad immediately before execution. Each command is given in full, with its complete pathname and with the values of arguments inserted.

-XOFF (D) Disable input examination.

-NX[ON] (D) Same as -XOFF.

EXAMPLES

1. \$ set -p1 'Input> '

Change the current Shell's primary prompt to "Input> ". Note the use of quotes to preserve the trailing blank.

2. \$ set -eon -xon -von Enable variable evaluation, command examination, and verification.

SH (SHELL) -- Invoke a Shell (command line interpreter).

FORMAT

SH [options] [pathname [arg ...]]

SH is a command line interpreter: it reads lines you type and interprets them as requests to execute other programs. For general information about the Shell, see Chapter 3. The DOMAIN System User's Guide also includes information on the Shell, particularly the method that it uses to process input.

By default, the Shell is running all the time. When you give the command SH, you generate a separate, subordinate Shell. This Shell can carry on separate operations, and can execute special programs and scripts containing command lines.

Note: The SH command does not create a new process, only a subordinate Shell running in the current process. Refer to the Display Manager section of this manual for information on the CP command for creating a new process.

ARGUMENTS

pathname

(optional)

Specify file containing a Shell script to be executed. Each line in the file will be interpreted as a Shell command.

Default if omitted: read standard input.

args

(optional)

Specify any arguments to be passed to the program in file 'pathname'. Arguments are substituted for 'n expressions in the program: arg1 for '1, arg2 for '2, etc. (See the DOMAIN System User's Guide for details on passing arguments to Shell commands.) See example 1 below.

Default if omitted: no arguments passed.

OPTIONS

Default options are indicated by "(D)."

-B[ON]

Send the output of a background process (created with the & parsing operator) to the display. The output of the background process is displayed in the transcript pad of the Shell where it was invoked. If you do not specify -B, the output of the background process is sent to /DEV/NULL.

- -BOFF
- (D) Do not display output from a background process.
- -NB[ON]
- (D) Same as -BOFF.
- -C[OMMAND] arg1 ...

Execute the following argument(s) as a Shell command, exactly

as if it had been read as an input line. If any argument contains explicit blanks, enclose it in quotes. The Shell passes all text following -C to 'arg1' as arguments, so if you want to specify other options to the SH command itself, they must precede -C.

-E[ON]

Enable evaluation of variables outside of expressions. If -E is specified, the Shell always evaluates variables, regardless of the context in which they appear. If -E is not specified, variables are evaluated only inside variable expression delimeters, ((expression)); otherwise, the Shell treats the `var_name expressions as strings and they are not evaluated.

- -EOFF (D) Evaluate variables only inside expressions.
- -NE[ON] (D) Same as -EOFF.
- **-F[IRST]** Do not exit after executing the command given by the -C option. This option is valid only if -C has been specified, and must precede -C on the command line.
- -I[NTER] Behave as though input is being entered interactively: prompt for each input line, and do not exit on errors or quit faults (DQ or CTRL/Q from keyboard). Normally, the Shell only executes interactively if its input comes from a pad or SIO line. Use of this option forces prompting.
- -S[CRIPT] (D) Behave as though executing a Shell script: do not prompt and abort on error. A Shell normally will not quit; any error or quit command is assumed to apply only to the last command given to the Shell.
- -NI[NTER] (D) Same as -S.
- -N[EXECUTE]

Interpret each command line only; suppress execution.

- -EX[ECUTE] (D) Interpret each command line and execute it.
- -P[ROMPT]1 prompt_string

Define the prompt string for the Shell created with SH.

-P[ROMPT]2 subprompt string

Define the subpromt string for the Shell created with SH. (The subprompt appears when you continue a Shell command over more than one line).

- -START [file] (D) Execute the specified script after the Shell is created. If 'file' is not specified, the Shell searches for a file called ~USER_DATA/SH/STARTUP and executes it if it exists. No error occurs if that file does not exist.
- -NSTART Disable startup file execution.
- -V[ON] Display each line of text in the transcript pad as it is read by the Shell program.

SH (SHELL)

- -VOFF (D) Disable input verification.
- -NV[ON] (D) Same as -VOFF.
- -X[ON] Display each command in the transcript pad immediately before execution. Each command is given in full, with its complete pathname and with the values of arguments inserted.
- **-XOFF** (D) Disable input examination.
- -NX[ON] (D) Same as -XOFF.

EXAMPLES

- 1. \$ sh program-name arg1 arg2 ...
- The Shell executes the commands in the file 'program-name', and substitutes the arguments ('argn') for character sequences ^n in the program file.

2. \$ sh -n my_script

Interpret each line in 'my_script',
but do not execute anything.

SIGP (SIGNAL PROCESS) -- Signal a Process.

FORMAT

SIGP [process_name ...] [options]

SIGP causes a quit or stop fault in a process. This is particularly useful for stopping background processes such as those created by the CPO (CREATE_PROCESS_ONLY) and CPS (CREATE_PROCESS_SERVER) Display Manager commands.

You may discover which processes are currently active by using the PST (PROCESS_STATUS) command.

ARGUMENTS

process name

(optional)

Specify name of process to be signalled. Multiple process names and wildcarding are permitted.

Default if omitted: -UID required (below)

OPTIONS

Default options are indicated by "(D)."

-Q[UIT] (D) Cause a quit fault in the process (like the Display Manager command DQ (CTRL/Q)). Executing programs halt, but the

process remains active.

-S[TOP] Ask the entire process to stop cleanly (closing streams, etc.).

-B[LAST] Stop the process in the nucleus (don't go to user mode). This

brings everything to a halt without letting the system attempt to

clean up.

-UID high low

-UID high.low Stop the process with the given UID. "high" and "low"

indicate the two halves of the UID.

-L List processes signaled.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1. \$ sigp process_7 -quit

Generate a quit fault in process_7, which will halt the program currently running there, but leave process_7 itself active.

SIGP (SIGNAL_PROCESS)

2. \$ sigp process_7 -stop -L Stop process_7 completely.
 "process_7" stopped.
\$

SIORF (SIO_RECEIVE_FILE) -- Receive a file from a remote host.

FORMAT

SIORF [options] [pathname ...] [*]

SIORF accepts remote host transmissions from the appropriate SIO line, decodes them according to the proper protocol, and writes them to the file you specify.

Arguments and options may appear in any order and are processed and take effect as encountered. This means options must be specified before the file(s) for which they are intended.

The transmission protocols used by SIORF are described in the SIOTF (SIO_TRANSMIT_FILE) command description.

ARGUMENTS

pathname
(ontional)

Specify name of file to receive the transmission. If you omit the filename, SIORF waits for the host to specify a receive file. (If you want the transmission written to standard output, use the "*" option.) SIORF terminates when it receives an end-of-transmission (EOT) signal, unless you include the -F option.

Default if omitted: see above

OPTIONS

-L n	Specify SIO line being used for the transmission. The default SIO line is 1.
-N	Select the Nibble protocol. (See "Protocols" following the SIOTF (SIO_TRANSMIT_FILE) command description.)
-F	Cause SIORF to monitor the SIO line for host transmissions until it receives an error message over the SIO line or CTRL-Q from the node. When you include this option, EOT does not cause SIORF to terminate.
-OBJ	Write transmissions to a DOMAIN object file. The SIO line used for the transmission must have been previously configured (using the TCTL command) in the following manner:

\$ tctl -noquit -sync -noinsync

to receive an object file, presumably from another node.

-R Replace file(s) if they already exist.

-AE Abort on error. Otherwise, transmission continues until EOT.

SIORF (SIO_RECEIVE_FILE)

-X host file

Request a remote host file to be transmitted. This presumes a host counterpart of SIOTF (SIO_TRANSMIT_FILE) is active.

非

Receive transmission to standard output. This option is valid only if the 'pathname' argument is omitted.

EXAMPLES

1. \$ tctl -line 2 -noquit -sync -noinsync Create (or replace) a binary \$ siorf -L 2 -r -obj prog.bin file PROG.BIN with the data

file PROG.BIN with the data received over SIO line 2; presumably being sent by an SIOTF counterpart.

2. \$ siorf -r -f

Receive files over SIO line 1
whose names are specified by
the transmission side (host
or DOMAIN node). Existing
files are replaced if needed.
SIORF remains active "forever"
until CTRL/Q or error occurs.

3. \$ siorf -x ask_file /eng/new_copy

Request file ASK_FILE to be sent over SIO line 1 and write data received to /ENG/NEW_COPY. Presumes the other side is running SIOTF or equivalent in "forever" mode. SIOTF (SIO_TRANSMIT_FILE) -- Transmit a file to a remote host.

FORMAT

SIOTF [options] [pathname ...] [*]

SIOTF sends the DOMAIN file(s) you specify to a remote computer ("host") using the appropriate SIO line and protocol.

Arguments and options may appear in any order and are processed and take effect as encountered. This means options must be specified before the file(s) for which they are intended.

ARGUMENTS

pathname
(optional)

Specify name of file to be transmitted. If you wish to transmit

data from standard input, use the "*" option.

Default if omitted: must use "*"

OPTIONS

-L n Specify SIO line to be used for transmission. The default SIO line is 1.

-N Select the Nibble protocol. (See the "Protocols" section below.)

-F Cause SIOTF to continue monitoring the SIO line for transmission requests from the remote host rather than

terminating when transmission is complete.

Transmit a DOMAIN object file (presumably to another node).

The SIO line used for the transmission must have been previously configured (using the TCTL command) in the

following manner:

\$ tctl -nosync -insync

-AE Abort on error rather than attempting to continue.

-X host_file Pass a filename to the remote host. The host can use this name for the next file it receives from the node. This presumes a host

for the next the reference from the next. This product

counterpart to SIORF (SIO_RECEIVE_FILE) is active.

Read from standard input and send standard input to the remote host. Signal end of data with an end-of-file (CTRL/Z).

EXAMPLES

1. \$ siotf -f

Wait for file requests over SIO line 1 and transmit them.

2. \$ tct1 -line 2 -nosync -insync \$ siotf -L 2 -obj prog1.bin prog2.bin Transmit file PROG1.BIN, then transmit file PROG2.BIN over SIO line 2. Presumably the receiving side is using SIORF

with -OBJ also, and has SIO line 2 configured properly.

3. \$ siotf -x tell_file /eng/notes

Send the name 'tell_file', then transmit the file /ENG/NOTES. Presumably the receiving side is in "forever" mode (-F) and thus waiting for instructions.

PROTOCOLS

To permit binary and ASCII file transmissions, we have implemented two protocols: Plain and Nibbled.

Plain

Plain protocol is the default. It assumes that the host operating system can transmit and receive all 256 bit patterns, so there is no need to use escapes or to nibble at the ASCII or binary files. Even if the host can handle only the ASCII character set, you should use the plain protocol for transmitting ASCII files. The format of this protocol is:

STX type COUNT...data...CHECKSUM CR

where:

STX

is the standard ASCII STX (02).

type

is a small ASCII letter that identifies the record type as follows:

a ACK d DATA e EOF h HELLO

g ANSWER HELLO

n NAK x NAME z EOT ? ERROR MESSAGE

COUNT

is the number of data bytes in the record (not to exceed 255), nibbled and transmitted as two ASCII bytes (@ and the capital letters A through O).

CHECKSUM

is a 1-byte calculated checksum, nibbled and transmitted as two ASCII bytes.

CR

is a standard ASCII carriage return.

Nibbled

If the host cannot send or receive anything but ASCII characters, use Nibbled protocol to transmit non-ASCII data. Transmitted records use a record format identical to that of Plain protocol, except that ~S replaces the STX byte. For SIOTF, each byte from the file is nibbled into two ASCII characters (@ or A through O). For SIORF, the low four bits of each two bytes received are concatenated; this protocol checks the ASCII range of the received bytes. A byte out of range causes SIORF to send the host an NAK signal. A byte out of range in five consecutive records causes SIORF to issue an error message, and terminate. The count field of nibbled records contains the original count (i.e., the number data bytes before nibbling). To select the Nibbled protocol, use the -N option with SIORF or SIOTF.

When you execute SIORF, it issues the HELLO record to signal that it is there, and to clear any transmission that may have preceded your command. It expects to receive the ANSWER HELLO response. SIOTF also does this before it begins transmitting records.

SIORF acknowledges each remote host transmission. SIOTF waits for the host to acknowledge each transmission. These acknowledgements have the format:

STX a CR (or) STX n CR

STX is either STX or ~S, and a and n are the small letters a (ACK) and n (NAK). The programs recover from a NAK by retransmitting the record in question. After ten consecutive unsuccessful retries, the programs will issue an error message and abort. All messages must be acknowledged, including error messages.

END-OF-FILE is signalled by a record of this format:

STX e CR

where e must be the small letter e. Host programs should acknowledge the EOF signal.

END-OF-TRANSMISSION is signalled by a record of this format:

STX z CR

where z must be the small letter z. Host programs should ACK the EOT signal. If the programs do not receive transmissions or ACKs for 60 seconds, they issue time-out error messages and terminate.

NOTES

SIOTF opens a stream to its SIO line in COOKed mode, SIORF opens the stream in RAW mode. Both programs synchronize with host XON/XOFF (CTRL/Q, CTRL/S) signals.

If you specify a DOMAIN file that cannot be opened, the programs issue an error message. If a file specified by a record received from the host cannot be opened (or created), the programs will issue an error message, and transmit an error message to the host. However, they will continue processing their parameters or (if you specified -F) waiting for host requests.

SIOTF does not transmit EOT if you specify -F. SIORF will not terminate at EOT if you specify -F. If you omit -F, SIORF will wait until it receives an EOT signal from the host, or times out.

The programs accept type "?" error messages instead of ACK or NAK signals. The programs display the error messages, and terminate (even if you specified -F). If SIORF gets an error message while receiving a file, it aborts. If you included -F, the programs try to remain active as long as possible.

Model programs to serve as the host-side counterparts to SIORF and SIOTF have been supplied in /SYS/SOURCE/EMT. These are models in FORTRAN and in Pascal. The FORTRAN subroutines that need to be modified for host-specific use are in HOST_MODEL_SUBSI.FTN. The Pascal procedures to be modified are clearly marked in the Pascal model programs. For a particular host environment, you may also need to modify other areas of these models.

SOURCE -- Execute a Shell script at the current Shell level.

FORMAT

SOURCE script_name [arg1...]

SOURCE allows you to execute a Shell script at the current Shell level. When you type

\$ MY_SCRIPT arg1

your script runs in a new (subordinate) Shell level. This means that all variables are now defined at a new level; that your script can't delete or otherwise effect variables at the level above; that state settings like VON/BON/EON that get set in the script vanish when the script finishes, and so forth.

On the other hand, typing

\$ SOURCE MY_SCRIPT arg1

executes the script at the current Shell level, just as though you had typed the contents of MY_SCRIPT into the process input window (and filled in the command line arguments yourself). If the script says 'von' then VON will be set after the script exits. If it defines a variable, that variable will still be defined, etc.

ARGUMENTS

script_name

(required)

Specify the name of the script to be executed.

arg1...

(optional)

Specify any arguments to be passed to the script.

Default if omitted: no arguments passed.

SRF (SORT_FILE) -- Sort and/or merge text files.

FORMAT

SRF [options] [pathname ...]

SRF sorts lines of all the named files together and writes the result on the standard output.

The sort key is an entire line. Default ordering is alphabetic by characters as they are represented in ASCII format (digits, then uppercase characters, then lowercase characters, then special characters).

ARGUMENTS

pathname

(optional)

Specify file(s) to be sorted. Multiple pathnames and wildcarding

are permitted.

Default if omitted: read standard input

OPTIONS

Sort Key Control: one of the following

-B Omit leading blas

-S n Sort based on the subfield starting in column n. If this option is

Omit leading blanks from keys.

omitted, sorting starts in column one.

-F Fold all letters to a single case.

Input Character Control: either of the following

-D Use 'dictionary' order: only letters, digits and blanks are

significant in comparisons. Special characters (punctuation,

control characters, etc.) are ignored.

-I Ignore all nonprinting, nonblank characters.

Sort Mode Control: one of the following

-M Merge only, the input files are already sorted.

-R Reverse the sense of the sort; list output entries in reverse order.

EXAMPLES

1. \$ 1d -c -dtm | srf

List contents of current working directory in order of date last modified.

- 2. \$ 1d -c -dtm | srf -r
- List most recently changed files first.
- 3. \$ 1d -c -b1 | srf -r
- List files by size with largest files first.

STCODE (STATUS_CODE)

STCODE (STATUS_CODE) -- Translate status code value to text message.

FORMAT

STCODE hex_stat_code

STCODE prints the text message associated with a hexadecimal status code. This command is useful when a user program produces a hexadecimal status code instead of the textual message.

STCODE processes pre-defined status codes. No provision is currently made to add user-defined status codes to the error text database.

ARGUMENTS

 ${\tt hex_stat_code}$

(required)

Specify hexadecimal status code to be translated.

EXAMPLES

\$ stcode 80001 disk not ready (from OS / disk manager) SUBS (SUBSYSTEM) -- Set or display subsystem attributes.

FORMAT

SUBS object [subs_name] [options]

SUBS is used to set or show protected subsystem attributes. When setting subsystem attributes, you must be running in that subsystem.

See the ENSUBS (ENTER SUBSYSTEM) command for details on entering a subsystem.

ARGUMENTS

object

(required)

Specify pathname of an object. The function of the object (either a protected file or a managing program) is determined by options described below.

subs_name

(optional)

Specify name of a subsystem. The Shell will search the directory /SYS/SUBSYS for the specified subsystem. If this argument is specified, the attributes of the named subsystem will be set as directed by the options described below.

Default if omitted: display attributes of 'object.'

OPTIONS

-DATA Set or display the name of the subsystem which manages

'object.'

-MGR Set or display the name of the subsystem for which 'object' is a

manager. 'Object' must be an executable file (i.e., a program).

-UP Increase the privilege level of a process running in a subsystem

so that it can directly access the objects it owns.

-DOWN Decrease the privilege level of a process; opposite of '-UP.'

-L List subsytem attributes and/or manager fields. This is the

default action if 'subs_name' is not specified.

-BR Display only the name of the subsystem. Not valid if attributes

are being set.

EXAMPLES

The following example illustrates the use of protected subsystems. First we show a Pascal source program written to "manage" the subsystem. (The calls issued to /SYS/INS/ACLM.INS.PAS to enable proper subsystem ACL checking are documented in the DOMAIN System Call Reference.) Following that is a Shell script that installs the subsystem using the CRSUBS, ENSUBS, and SUBS commands.

```
Pascal Source Manager
{ pse --- protected subsystems example program }
{ usage:
           pse pse_file out_file
    where:
        pse_file
                        protected object owned by 'PS EXAMPLE' subsystem
        out file
                        output file
    The 'PSE' program is used to extract the protected data from
    objects owned by the 'PS EXAMPLE' subsystem and put them in an
    output file. As a trivial example, the protected data has
    a sequence number in the first 8 columns of each line, which
    is not logically part of the data, but which can be imagined
    to be important to the integrity of the data. Extracting the
    data removes the sequence number and copies the rest of the
    line to the output file. If this were a real application, it
    might also format and/or select the data sent to the output file.
}
program pse;
%include '/sys/ins/base.ins.pas';
%include '/sys/ins/streams.ins.pas';
%include '/sys/ins/error.ins.pas';
%include '/sys/ins/pgm.ins.pas';
%include '/sys/ins/aclm.ins.pas';
type
   buf_t = array[1..128] of char;
var
   istrid: stream $id t;
                                   { input stream ID }
   ostrid: stream $id t;
                                   { output stream ID }
   arg:
          name_$pname_t;
                                  { command line argument }
   alen: integer;
                                  { length of command line argument }
           status $t;
   st:
                                   { status code }
   sk:
           stream $sk t;
                               { stream seek key }
   buf: buf_t;
                                   { I/O buffer }
           ^buf_t;
   bp:
                                   { pointer to same }
   blen: integer32;
                                   { length of I/O buffer }
    { get input file name }
   alen := pgm_$get_arg(1, arg, st, sizeof(arg));
   if (st.code <> 0) then begin
       writeln('input file name missing.');
       error $print(st);
       pgm $set severity(pgm $error);
       pgm $exit
       end:
   { open input file; must increase privilege to access
       my own protected file... }
   aclm $up;
                                               { get more privilege }
   stream_$open(arg, alen, stream_$read,
      stream_$no_conc_write, istrid, st);
   aclm $down;
                                               { decrease privilege }
   if (st.code <> 0) then begin
       writeln('Can''t open input file.');
```

error \$print name(st, arg, alen);

```
pgm $set severity(pgm $error);
        pgm_$exit
        end;
    { get output file name }
    alen := pgm_$get_arg(2, arg, st, sizeof(arg));
    if (st.code <> 0) then begin
        writeln('output file name missing.');
        error $print(st);
        pgm $set severity(pgm $error);
        pgm_$exit
        end;
    { create output file; DO NOT increase privilege: it would
        be an error to write on one of my own protected objects
        here -- I want an ordinary file }
    stream_$create(arg, alen, stream_$overwrite,
       stream $no conc write, ostrid, st);
    if (st.code <> 0) then begin
        writeln('Can''t create output file.');
        error_$print_name(st, arg, alen);
        pgm_$set_severity(pgm_$error);
        pgm $exit
        end;
    { now just copy the file ... a real program would be more
        complicated here. }
    repeat
        { read a record... }
        aclm_$up;
        stream $get rec(istrid, addr(buf), 128, bp, blen, sk, st);
        aclm $down;
        if st.code <> 0 then
                                    { error or EOF }
        { write a record, stripping off the sequence number.
            Notice I did NOT make a check to see that the length
            of the record was greater than 8 characters... I am
            confident that the rest of the subsystem correctly
            maintains sequence numbers, and that the protected
            subsystem mechanism makes sure that only the subsystem
            can operate on the data. }
        stream_$put_rec(ostrid, addr(bp^[9]), blen-8, sk, st);
        until st.code <> 0;
    { check that we stopped because of EOF }
    if (st.subsys = stream $subs) and then
      (st.code = stream $end of file) then
        pgm_$exit;
    { not EOF --- a real error of some sort, then }
    writeln('I/O error: ');
    error_$print(st);
    pgm_$set_severity(pgm_$error);
   pgm_$exit
end.
```

Shell Script

```
# create a protected subsystem
crsubs ps example
     # create some data to be protected --- normally a special create
     # operation would be used that guarantees data integrity
catf >ps_data <<!
12345678this is some protected data
12345679next record of protected data
ensubs ps_example -v <<!
                                 # enter the new subsystem in a shell
subs pse ps_example -mgr
                                 # make pse a manager of 'PS EXAMPLE'
subs ps_data ps_example -data
                                 # protect the data
edacl ps_data -d % -a %.backup r # now can only get at data from within
subs -up
cpf ps_data ps_data2 -subs
                                  # make a copy of the data
subs -down
pse ps data out file
                                 # run PSE to extract the protected data
catf out file
                                 # see the protected data
     # now see how it fails if I try to make the output file a
     # protected object of the 'PS EXAMPLE' subsystem...
pse ps_data ps_data2
                                 # try to clobber ps_data2
```

TB (TRACEBACK) -- Print traceback after a fault.

FORMAT

TB

TB prints a traceback after a fault. The traceback lists the names of the routines leading from the fault back to the main program, and includes the line number at which each routine was called. Up to 32 levels of calls can appear. If the sequence exceeds 32 calls, the first 16 and last 16 are shown in the traceback.

TB requires no arguments or options.

NOTE: There is a homonymous DM command: TB -- To bottom of window. See the TB command description in the DM chapter for details.

EXAMPLES

The traceback from a floating-point error might resemble the following:

```
$ TB
overflow in multiply (from library/floating point)
In routine "EXTRAPOLATE" line 25
Called from "REFLECTION" line 192
Called from "$MAIN" line 19
```

In this example, an overflow error occurred in a floating-point multiplication at line 25 of the routine named EXTRAPOLATE. The routine EXTRAPOLATE was called at line 192 of the routine REFLECTION, which in turn was called at line 19 of the main program.

TCTL (TERMINAL_CONTROL) -- Set or display SIO line characteristics.

FORMAT

TCTL [options]

TCTL sets or displays SIO line characteristics, which control how hardware and software connected to those lines should behave. For example, if you wish to allow a dumb terminal to dial into a node and communicate meaningfully with a Shell, you must properly configure the SIO line that the terminal will use so that the node will understand the terminal's signals. Thus TCTL controls the transmission speed (baud rate) that connected terminals must use, and which characters typed on those terminals delete characters or lines.

OPTIONS

If no options are specified, the current settings of the SIO lines are displayed.

-DEFAULT Set all settable options to their default values. This allows you

to quickly reset values to known states.

-LINE n Specify the SIO line to be affected by subsequent options on this command line. 'n' is an integer in the range 0-3. The default SIO

line is line 1 or standard input (if standard input is directed to

an SIO line).

-SPEED baud

Set the speed of the line, for both input and output. The possible baud rates are: 50, 75, 110, 134, 150, 300, 600, 1200, 2000, 2400, 3600, 4800, 7200, 9600, 19200. The initial setting is 9600 baud. Note that 3600 baud is not supported on DN3xx systems. Speeds for partner line(s) may occasionally need to be

forced: see -FORCE below.

-FORCE Valid only if -SPEED is also specified. This option forces the

speed of the line specified by -LINE to be set to the correct speed (specified by -SPEED). If the line has a "partner line" that is currently set to some other (incompatible) speed, -FORCE will reset the partner line's speed to 9600 baud. See example 4 below. For more information about partner lines, see the SIO_\$CONTROL description in Programming with General

System Calls.

-NLD [n] Set NEWLINE delay. This is the number of milliseconds

required following the output of a line feed (NEWLINE). If 'n' is

omitted or not set, 20 milliseconds is the default.

-ERASE char Set the erase character. This option is valid only when data is

being passed to the SIO line in "cooked" mode. 'char' may be any character or a one-byte hexadecimal value. Some characters may require quoting in the Shell. The erase character is initially set to BACKSPACE (08 hex).

-KILL char

Set the kill character. This option is valid only when data is being passed to the SIO line in "cooked" mode. The kill character is initially set to CTRL/X.

-EOF char

Set the end-of file character. The EOF character is initially set to CTRL/Z.

-QUITCHR char

Set the quit character. The quit character is initially set to CTRL/].

-INTCHR char

Set the interrupt character. This is used primarily by DOMAIN/IX. The interrupt character is initially CTRL/C.

-SUSPCHR char

Set the suspend character. This is used primarily by DOMAIN/IX. The suspend character is initially CTRL/P.

-[NO]RAW

Turn raw mode on or off. In raw mode, full 8-bit bytes are transmitted in both directions, without any interpretation. The initial setting is -NORAW.

-[NO]ECHO

Turn the echoing of input characters over the SIO line on or off. The initial setting is ECHO.

-[NO]SYNC

The terminal normally sends CTRL/S (XON) when its input buffer begins to fill, and CTRL/Q (XOFF) when the buffer begins to empty, to synchronize it with a high-speed data source. This option enables or disables that behavior (it is initially enabled). -SYNC implies -NORTS_ENABLE.

-[NO]CVT_NL

Enable or disable conversion of LF to CR-LF on output. CVT_NL causes NEWLINES (LF) to be transmitted as CR-LF sequences. This option is valid only when data is being passed to the SIO line in "cooked" mode. The initial setting is -NOCVT_NL. NOTE: EMT always puts the SIO line in "raw" mode, so -CVT_NL has no effect in that instance. Use the OUTTERM command within EMT.

-[NO]CVTRAW_NL

Similar to -CVT NL, but applies only to raw mode.

-[NO]INSYNC

Enable or disable reacting to CTRL/S and CTRL/Q when received by node. -INSYNC causes transmissions to halt when CTRL/S is received and resume when CTRL/Q is received. The initial setting is -NOINSYNC.

-PARITY state

Select parity checking state. Valid states are:

NONE don't send or check parity bit
EVEN send and check even parity
ODD send and check odd parity

The initial state is NONE.

-BPC n Set number of bits per character. 'n' is an integer in the range

5-8. The initial number of bits per character is 8.

-STOP n Set number of stop bits. 'n' may be 1, 1.5, or 2. The initial

number of stop bits is 1.

-[NO]QUIT Enable/disable quits for the current process. The initial setting

is -NOQUIT.

-[NO]INT Enable/disable interrupts for the current process. The initial

setting is -NOINT.

-[NO]SUSP Enable/disable suspend faults for the current process. The

initial setting is -NOSUSP.

-[NO]RTS Enable/disable the request-to-send line. The initial setting is

-RTS. Note that you may NOT use this option if

-RTS_ENABLE is specified.

-[NO]DTR Enable/disable the data-terminal-ready line. The initial setting

is -DTR. Note that -DTR is not valid if -LINE 3 is specified.

-[NO]DCD_ENABLE

Enable/disable standard handling of carrier detect. The initial

setting is -NODCD ENABLE.

-[NO]CTS_ENABLE

Enable/disable standard handling of clear-to-send. The initial

setting is -NOCTS_ENABLE.

-[NO]RTS_ENABLE

Enable/disable RTS flow control. The initial setting is

-NORTS_ENABLE. Enable implies -NOSYNC.

-[NO]BP_ENABLE

Enable/disable processing of bit-pad input (from a graphics tablet) on the SIO line. When enabled, data received on this line is not delivered through STREAM_\$GET_REC, but is accumulated by the interrupt routine, and passed to the display driver a point at a time, much as with the touchpad. This processing has the additional property that subsequent points within +/-1 in both the X and Y dimensions are ignored. The initial setting is

-NOBP_ENABLE.

-ERROR state

Select error reporting state. Valid states are:

[NO]FRAMING enable/disable reported framing errors enable/disable reported parity errors [NO]PARITY [NO]DCD_CHANGE enable/disable report on DCD line [NO]CTS CHANGE enable/disable report on CTS line

Only FRAMING is initially enabled.

EXAMPLES

1. \$ tctl

Status of Line 1:

Display current settings.

Erase (character delete) character: 08 (hex)

Kill (line delete) character: 18 (hex)

End of file character: 1A (hex)

Quit character: 1D (hex) Interrupt character: 03 (hex) Suspend character: 10 (hex)

New line delay: 0 Speed: 9600

Raw: FALSE,

Echo: TRUE,

Cvt NL: TRUE

CvtRaw_NL: FALSE, Host_Sync: TRUE,

Input Sync: FALSE

DTR: TRUE,

DCD: FALSE

RTS: TRUE, CTS: FALSE,

Quit_Enable: FALSE, Int_Enable: FALSE

Susp_Enable: FALSE, DCD_Enable: FALSE, CTS_enable: FALSE

BP enable: FALSE RTS enable: FALSE

Eight bits per character, Parity: None, One stop bit

Errors enabled: FRAMING

2. \$ tctl -line 2 -quitchar OFE -insync -speed 300

Set quit character to hex FE, enable input synchronization, set speed to 300 baud on SIO line 2.

3. \$ tctl -parity odd -quitchar '#' -kill !

Set parity to odd, quit character to # (quoted because # normally begins a comment in the Shell), and kill character to ! on line 1.

4. \$ tctl -line 2 -speed 50

?(tctl) Speed requested is incompatible with current speed of partner line 1. Resubmit command with -FORCE if permissable to reset partner line to 9600 baud.

\$ tctl -line 2 -speed 50 -force

?(tctl) Warning: Speed of partner line has been reset to 9600 baud.

TEE -- Copy input to output and to named files.

FORMAT

TEE pathname ...

TEE copies its standard input to standard output and to the named files. It is useful for saving the data being transmitted through a pipeline.

ARGUMENTS

pathname

(required)

Specify name of file to receive output. Multiple pathnames are permitted.

EXAMPLES

\$ FMT mary | TEE mary.clean | OS >mary.overstruck

This command line causes the file mary to be formatted with FMT. The formatted text is written to the file mary.clean and also piped to the OS command to produce overstruck output (for a line printer) redirected into the file mary.overstruck. Thus, you end up with two output files: one with ASCII carriage control (mary.clean) and one with FORTRAN carriage control (mary.overstruck).

TLC (TRANSLITERATE_CHARACTERS) -- Replace characters.

FORMAT

TLC from-chars [to-chars]

TLC copies standard input to standard output, substituting or deleting selected characters. Each input character found in 'from-chars' is replaced by the corresponding character of 'to-chars'.

TLC differs from CHPAT (CHANGE_PATTERN) in that it deals only with single characters or ranges of characters, whereas CHPAT deals with character strings. For example,

\$ tlc xy yx

changes all x's into y's and all y's into x's, whereas

\$ chpat xy yx

changes all the patterns "xy" into "yx".

ARGUMENTS

from-chars

(required)

Specify existing character(s) to be replaced. You may specify a range of characters by separating the extremes with a dash. For example, a-z stands for the list of lowercase letters. 'from-chars' may contain a maximum of 100 characters.

to-chars
(optional)

Specify replacement characters. You may specify a range of characters by separating the extremes with a dash. For example, a-z stands for the list of lowercase letters. 'to-chars' may contain a maximum of 100 characters.

If 'from-chars' and 'to-chars' contain an equal number of characters, TLC translates the first character in 'from-chars' to the first character in 'to-chars', and so forth.

If 'from-chars' contains more characters than 'to-chars', TLC repeats the last character in 'to-chars' until 'to-chars' is as long as 'from-chars'. However, in the output, adjacent repetitions of the last character appear as one character. (See example 2 below.)

If 'to-chars' contains more characters than 'from-chars', the extra characters are ignored.

Default if omitted: delete all occurrences of characters in the 'from-chars' list.

EXAMPLES

The following examples show TLC's operation using standard input and output. The first line following the command line is an echo of standard input. The next line is the TLC results, then another line of input, then more results, and so forth.

- 1. \$ TLC te zq
 Now is the time
 Now is zhq zimq
 *** EOF ***
 \$
- 2. \$ TLC abc zq

Now is the time for all good men and boys to come to the aid Now is the time for zll good men znd qoys to qome to the zid abcaccbaa

zqzqzz aaaaa zzzzz bbbbb q ccccc q

*** EOF ***

Note that multiple occurrences of "a" are replaced by "z" one for one, but multiple occurrences of "b" and "c" are replaced with a single "q", since the 'from-char' list is longer than the 'to-char' list.

3. TLC A-Z a-z <mary.caps >mary.lc

This command changes all uppercase letters in the input file "mary.caps" to lowercase and writes the results to the file "mary.lc". Lowercase characters already in "mary.caps" remain unchanged.

TPM (TOUCH_PAD_MODE) -- Set/display touchpad and mouse characteristics.

FORMAT

TPM [options]

TPM allows you to define characteristics for the touchpad and mouse. The touchpad operates in one of three modes: absolute, relative, and absolute/relative. The mode of operation establishes how movements of your finger on the touchpad affect the position of the cursor on the screen. The three modes differ primarily in how the cursor moves when you lift your finger from the touchpad and then replace it. The subsections below describe the three operational modes, as well as the other options.

The mouse operates in relative mode only.

OPTIONS

If no options are specified, TPM displays the current touchpad characteristics.

Default options are indicated by "(D)."

-A	(D)	Select absolute mode.	
-R		Select relative mode.	

-AR	Select absolute/relative mode.
-----	--------------------------------

-RERANGE	Set prescaling factors for touchpad data.	
----------	---	--

-S x y	Set scaling factors for x and y. Values must be in raster units,
	and can range from 1 to 1024. The default scaling factors are
	799 for x and 1023 for y (portrait displays); and 1023 for x and
	799 for y (landscape displays).

-О х у	Set x and y as the origin for absolute mode. Values must be in
	raster units, and can range from 0 to 1023. The default origin is
	0,0.

-H n	Set the hyeteresis how size	The value must be in raster units,
	and can range from 0 to 1023	B. The default is 5.

Absolute Mode

In absolute mode, using the default scale and origin, the touchpad approximates the screen, so that the top left edge of the touchpad represents cursor positions at the top left edge of the screen. Absolute mode is the default setting. When you place your finger on the touchpad, the cursor jumps to a corresponding position on the screen. Moving your finger across the touchpad moves the cursor across the screen in the same direction.

For example, moving your finger from the top of the touchpad to the bottom moves the cursor

TPM (TOUCH_PAD_MODE)

from top to bottom on the screen. If you lift your finger from the touchpad, and later touch the pad again, the cursor jumps to a new position on the screen corresponding to the new finger position.

Absolute mode has no meaning if you are using a mouse.

Relative Mode

In relative mode, cursor movements correspond only to finger movements across the touchpad. The cursor does *not* move when you first place your finger on the touchpad. This differs from absolute mode, where the cursor jumps to a new position when you lift your finger and then replace it. In effect, relative mode causes the touchpad to correspond to different areas of the screen, relative to the current cursor position.

This is the only meaningful mode for a mouse: all movement begins from the current cursor position.

Relative mode is typically used with scale factors less than the defaults. Smaller scale factors mean that the touchpad maps to a smaller area of the screen. For example, scale factors of 200 by 256 specify one-sixteenth of a portrait display's screen area. With small scale factors, relative mode allows fine resolution of the cursor position within a small area.

To reach distant areas on the screen, you can use several "strokes" on the touchpad or mouse, each stroke moving the cursor closer to its final destination. To assist you in making large movements in relative mode without having to use too many strokes, the speed of cursor movement is artificially accelerated in relation to the speed of finger or mouse movement. Thus, a quick motion will move the cursor farther than a slow, deliberate motion which covers the same distance.

Absolute/Relative Mode

Absolute/relative mode is a combination of absolute and relative modes. It has no meaning for the mouse. In this mode, the first position of your finger on the touchpad establishes the first position of the cursor, as in absolute mode. Moving your finger across the touchpad moves the cursor across the screen. As in relative mode, the scale is typically smaller than the whole screen.

Unlike absolute and relative modes, however, the effect of lifting your finger from the touchpad depends on how long you break contact. If you lift and replace your finger quickly -- within a half second -- the cursor does not move, and the effect is the same as relative mode. If you break contact for more than a half second, however, the cursor jumps to a new absolute position when you put your finger on the touchpad again.

Absolute/relative mode is useful for "jumping" the cursor from one place to another, then carefully positioning it in the new area. For example, this mode is commonly used to move the cursor in a jump from one window to another, and then point to a character in the second window.

Prescaling the Touchpad

Raw touchpad data vary slightly from one touchpad to another. Prescaling is, in essence, calibration of the touchpad. Every time you start the node, the touchpad manager prescales the data to determine an exact range for the device.

To prescale, the touchpad manager observes the first thousand points of touchpad data (about 30 seconds of use). During this time, you should try to touch all four edges of the touchpad to ensure that the observed data constitute an accurate sample. Based on the observed data, the touchpad manager computes a prescaling factor which, when applied to the data, brings all points into the range from -.05 to 1.05. This range corresponds to the edges of the screen, plus an overlap of 5%, when multiplied by the default scaling factors. Because of the overlap, you need not touch the internal frame (under the conductive material) to move the cursor to the edge of the screen.

The -RERANGE option invokes prescaling. This option is useful if the first 30 seconds of use did not include the entire range of the touchpad. It is also handy if you change keyboards on a node, and therefore need to reset the prescaling factors without restarting the node.

Scale Factors

The touchpad manager translates, or scales, the data into raster units, which the Display Manager understands. Scale factors, specified with the -S option, are applied to the prescaled touchpad data to translate it to raster units for the Display Manager.

The scale factors are multiplied by the prescaled data. The default scale factors are 800 for x and 1024 for y (portrait displays); and 1024 for x and 800 for y (landscape displays). Applying these factors to prescaled data results in numbers from approximately 0 to 799 (for x) and 0 to 1023 (for y) for portrait displays, and vice versa for landscape displays. (Note that the prescaled data allow a 5% overlap, as mentioned in the preceding subsection.)

The default scale factors provide for touchpad data corresponding to the whole screen. Smaller scale factors reduce the area to which the touchpad maps, thereby allowing you to more finely tune the cursor position. This also applies to mouse movement, allowing changes in the apparent motion sensitivity of the device.

Setting the Origin

The origin is the point denoted by the upper left corner of the touchpad, in absolute and absolute/relative mode. In relative mode, the origin has no meaning. By default, the touchpad origin corresponds to the upper left corner of the screen, that is, the point 0,0 in raster units. By changing the origin, you can use the touchpad (in absolute mode) to correspond to a portion of the screen.

This feature is useful for applications that need to move the cursor within a fixed window, rather than across the whole screen. For example, a program that displays a menu in one window might set the origin to the upper left corner of the menu window. Consequently, the touchpad maps onto the menu window instead of the entire screen.

Hysteresis

The hysteresis value defines the dimensions of a "box" around your finger position on the touchpad or the current position of the mouse. Movement within the box does not change the position of the cursor on the screen.

Specify the hysteresis value in raster units. The touchpad manager compares the value to the difference between the current and previous finger positions on the touchpad or the current and previous positions of the mouse. If the difference is less than the hysteresis value, the cursor does not move. If the difference is greater than the hysteresis value, the hysteresis value is subtracted from the difference and the cursor moves the resulting distance. The default hysteresis value is five.

EXAMPLES

\$ TPM

Mode: absolute

Xscale: 1024, Yscale: 800

Hysteresis: 5 Origin: 0, 0

\$ TPM -ar -s 400 512

Display current characteristics

Set characteristics to absolute/relative mode with half the default scaling sensitivity (portrait display).

TZ (TIMEZONE) -- Set or display system time zone.

FORMAT

TZ [ts_name | utc_delta [new_ts]]

TZ sets the system time zone to a known time zone or to an offset from Coordinate Universal Time (UTC).

To set the actual time registered by the nodes's internal clock, use the CALENDAR command. See the appendices for more information.

ARGUMENTS

If no arguments are specified, TZ displays the current setting.

tz_name (optional)

Specify new time zone. Valid names are:

Name	Time Zone
EDT	Eastern Daylight Time
EST	Eastern Standard Time
CDT	Central Daylight Time
CST	Central Standard Time
MDT	Mountain Daylight Time
MST	Mountain Standard Time
PDT	Pacific Daylight Time
PST	Pacific Standard Time
GMT	Greenwich Mean Time
UTC	Coordinated Universal Time

Default if omitted: use 'utc_delta' argument

utc_delta (optional)

Specify positive or negative offset from UTC. The plus sign is optional for positive offsets. Format for offset is hh:mm (e.g., -10:00 for ten hours earlier than (west of) Coordinated Universal Time). Only whole or half hour offsets may be specified. Other fractional offsets produce an error message.

Default if omitted: use 'tz_name' argument

new_tz
(optional)

Specify new time zone name to be assigned to the zone indicated by the 'utc_delta' argument. Use this argument to create time zones that are not included in the list above.

Default if omitted: no name assigned

TZ (TIMEZONE)

EXAMPLES

\$ TZ

Timezone: EST

Delta from UTC: -5:00

\$ TZ pdt

Set time zone to Pacific Daylight Time

Display current time zone.

\$ TZ 4:30 gst

Create (and set) a time zone named GST that is four and a half hours later than (east of) Coordinated Universal Time.

UCTNODE (UNCATALOG_NODE) -- Uncatalog a node.

FORMAT

UCTNODE pathname ... [options]

UCTNODE removes the specified entry directory name from the local copy of the network root directory. After the name is removed, objects cataloged under that node's entry directory are no longer accessible to you or other nodes on the network.

Node entry directories are created with the CTNODE (CATALOG_NODE) command.

ARGUMENTS

pathname

(required)

Specify node entry directory name to be uncataloged. Multiple pathnames and wildcarding are permitted.

OPTIONS

-L

List directory names as they are uncataloged.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ uctnode als_node

Uncatalog the node with the entry directory name specified.

UCTOB (UNCATALOG_OBJECT) -- Uncatalog the specified pathname, without deleting the associated object.

FORMAT

UCTOB pathname ... [options]

UCTOB removes the specified pathname from the name space. The object associated with the pathname is not affected. This command is primarily intended for system-level debugging use.

ARGUMENTS

pathname

(required)

Specify name of object to be uncataloged. The object itself is not affected. Multiple objects and wildcarding are permitted.

OPTIONS

-BR

Suppress listing of names and UIDs of objects as they are uncataloged. These are reported unless this option is specified.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

\$ uctob testfile
"testfile" uid is 16791COC.40000074.

Uncatalog "testfile".

ULKOB (UNLOCK_OBJECT) -- Unlock an object.

FORMAT

ULKOB [pathname ...] [options]

The ULKOB command unlocks objects residing on, or locked by processes running on, the current node. You may not unlock objects on remote nodes unless you locked them (see -F below). This command can be used when a program terminates abnormally, leaving objects locked, or to unlock objects previously locked with the LKOB (LOCK_OBJECT) command.

To obtain a list of your node's locked objects, use the LLKOB (LIST_LOCKED_OBJECTS) command.

ARGUMENTS

pathname

(optional)

Specify name of object to be unlocked. Multiple pathnames and

wildcarding are permitted.

Default if omitted: -U option must be specified

OPTIONS

If no options are specified, the object is unlocked for all lock modes.

-R Unlock an object that was locked for read mode; the lock must

be owned by this process.

-W Unlock an object that was locked for write mode; the lock must

be owned by this process.

-I Unlock an object that was locked for reading with intent to

write; the lock must be owned by this process.

-F Forcibly unlock an object. It may have been locked for any

mode and the lock may be owned by any process. The object must reside on the current node, however, or must have been locked by the current node. In other words, you cannot unlock

objects on a remote node unless you locked them.

-L List the name of each object as it is unlocked.

-U uid ... Specify the UID of the object(s) to unlock. Multiple UIDs are

permitted. If the 'pathname argument is omitted, then this

option is required.

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

ULKOB (UNLOCK_OBJECT)

EXAMPLES

- \$ ULKOB mary -f Forcibly unlock the file "mary" for any mode.
- \$ ULKOB -uid 1C1A9E2F.20000246 1C1A9E42.50000246
 Unlock the two objects with the specified UIDs.

VCTL (VT100_CONTROL) -- Set/display VT100 terminal characteristics.

FORMAT

VCTL [options]

VCTL allows you to set or display information about how the VT100 terminal emulator driver handles input from the keyboard (for example, whether or not it echoes characters, or how it interprets key sequences typed at the keyboard).

This command is valid only if you have the VT100 terminal emulation software package running on your node. In addition, VCTL can only be run in a window where the VT100 emulator is already running.

OPTIONS

If no options are specified, the current VT100 settings are displayed.

-DEFAULT

Set all options to their default values. This allows you to quickly reset values to known states.

-[NO]CVT_IN_NL

Convert a newline (linefeed) to a carriage return on input. The initial setting is -NOCVT_IN_LINE.

-[NO]CVT_IN_CR

Convert a carriage return to a newline on input. The initial setting is -CVT_IN_CR.

-[NO]CVT_OUT_NL

Convert a newline to carriage return, newline on output. The initial setting is -CVT_OUT_NL.

-[NO]CVT OUT CR

Convert a carriage return to a newline on output. The initial setting is -NOCVT_OUT_CR.

-[NO]ECHO

Turn the echoing of input characters on or off. The initial setting is ECHO.

-[NO]ECHO_CTL

Turn the echoing of control characters (such as CTRL/Z) on or off. The initial setting is NOECHO CTL.

-[NO]ECHO_ERASE

If ECHO is on, controls whether characters are visibly erased from the screen when the erase character is typed. The combination of ECHO and NOECHO_ERASE causes the erase character to be echoed until all characters on a line are erased. The initial setting is -ECHO_ERASE.

-[NO]ECHO_KILL

If ECHO is on, controls whether a line is visibly erased from the screen when the line kill character is typed. The combination of ECHO and NOECHO_KILL causes the kill character to be echoed and a new line to be displayed. The initial setting is -ECHO_KILL.

-EOF char

Set the end-of-file character. The EOF character is initially set to CTRL/Z.

-ERASE char

Set the erase character. This option is valid only when data is being passed to the terminal emulator in "cooked" mode. The 'char' can be any character or one-byte hexadecimal value. Some characters may require quoting in the Shell. The erase character is initially set to BACKSPACE (08 hex).

-INTR char

Set the interrupt character, which sends an interrupt fault to the process group of the terminal emulator. The interrupt character is initially set to CTRL/C.

-KILL char

Set the kill character. This option is valid only when data is being passed to the emulator in "cooked mode". The kill character is initially set to CTRL/X.

-QUIT char

Set the quit character. The quit character is initially set to CTRL/Q.

EXAMPLES

1. \$ vctl Display current settings.

Erase (character delete) character: "^H" (08 hex)

Kill (line delete) character: "~U" (15 hex)

End of file character: "^Z" (1A hex)
Interrupt character: "^C" (03 hex)
Quit character: "^Q" (11 hex)

Extra break character: FF (hex)
Raw: FALSE, Echo: T

Raw: FALSE, Echo: TRUE, Echo_Kill: TRUE, Echo_Ctl: FALSE, Cvt In NL: FALSE, Cvt Out NL: TRUE,

Echo_Erase: TRUE
Cvt_In_CR: TRUE
Cvt_Out_CR: FALSE

\$

2. \$ vctl -quit OFE -cvt_out_cr

Enable Sigs: TRUE

Set quit character to hex FE, enable conversion of output newlines to carriage returns.

VOFF -- Deactivate the Shell's -V flag.

FORMAT

VOFF

VOFF turns off the Shell's -V (Verify) flag, which is turned on by the VON command or the -V option on the SH command line. When the flag is off, command lines are not displayed when they are read by the Shell. Verification is off by default.

VOFF requires no arguments or options.

VON -- Activate the Shell's -V flag.

FORMAT

VON

VON turns on input verification. As commands are executed, or comments processed, they are written to the error output stream of the Shell. In Shell scripts, VON can be used to show the progress being made by the script.

If VON is turned on in a shell script, it remains on until that shell script exits, or until over-ridden by a VOFF in a nested shell script. When a shell script exits, the state of input verification is returned to the state in effect just before the script was invoked.

VON requires no arguments or options.

VSIZE (VT100_SIZE) -- Set/display VT100 window settings.

FORMAT

VSIZE [options]

The VSIZE command allows you to set the dimensions of the VT100 emulator window pane. This command is only valid from within the VT100 emulator (which is invoked with the VT100 command); attempting to use it directly from the Shell causes an error.

OPTIONS

If no options are specified, VSIZE displays the current window pane settings.

-L n Specify the height of the window pane in lines. If this option is omitted, the height remains unchanged.

-C n Specify the width of the window in columns. If this option is

omitted, the width remains unchanged.

EXAMPLES

\$ vt100 Invoke VT100 emulator.
\$ vsize Display current settings.
Old screen size is 18 lines by 70 columns.
New screen size is 18 lines by 70 columns.
\$ vsize -c 60 Change the width.
Old screen size is 18 lines by 70 columns.
New screen size is 18 lines by 70 columns.

New screen size is 18 lines by 60 columns.
\$ *** EOF ***
Exit the emulator and return
to the Shell.

VT100 -- VT100 terminal emulator.

FORMAT

VT100 [options] [pathname [arg1 arg2 ...]]

The VT100 command creates a window running the VT100 terminal emulator and starts up a Shell within the window. This command is valid only if you have the VT100 terminal emulation software package running on your node.

The VT100 terminal emulation package is intended for use with two types of programs. When used in conjunction with remote communications packages such as DOMAIN TCP/IP or X.25, the VT100 terminal emulator allows you to interact with the remote system as if you were logged into a VT100 connected to that system. Using the VT100 terminal emulator with programs that take advantage of VT100 special features allows you to run these programs on a DOMAIN node without having to tailor them to the DOMAIN environment.

The VT100 terminal emulation package consists of:

- The terminal emulation software, which performs the functions of a VT100 terminal, such as handling VT100-type escape sequences. The terminal emulator redirects the handling of keyboard input and screen output to stream manager operations. The terminal emulator is invoked within a DM window by the VT100 Shell command. Note that you cannot change the size of a DM window once the VT100 emulator is running within it.
- The terminal emulator driver, which performs keyboard input functions such as erasing or echoing characters. The VCTL Shell command allows you to set and display the VT100 terminal characteristics controlled by the terminal emulator driver.

ARGUMENTS

If any options are specified, they must precede the argument(s).

pathname [arg1 arg2 ...]

(optional)

Specify the name of a command or program for the Shell in the VT100 window to invoke. You must give the full pathname; for example, /com/ld. Arg1, arg2, ... are valid arguments to the selected command (or program): for example, /com/ld //my_node/my_home dir.

Default if omitted: invoke /com/sh

OPTIONS

If any options are specified, they must precede the argument(s). Once VT100 is running, you may change the window size with the VSIZE (VT100_SIZE) command.

-STD

Set up a VT100 window that is 24 lines by 80 columns (the standard size of a VT100 screen).

-LINES n

Set up a VT100 window with the number of lines specified by 'n'. The number of lines cannot be greater than the number of lines in the DM window running the VT100 emulator.

-COLUMNS n

Set up a VT100 window with the number of columns specified by 'n'. The number of columns cannot exceed the number of columns of the DM window running the VT100 emulator.

EXAMPLES

1. \$ VT100

Create a window running the VT100 emulator and start a shell running within the window.

2. \$ VT100 /COM/TELNET hostname

Open a connection to the remote system specified by 'hostname' and create a window running the VT100 emulator.

VT100 Keyboard Layout

The table below shows how the keys on a DOMAIN low-profile or 880 keyboard map to the keys of a VT100. This presupposes that you are running the VT100 Keyboard Emulation package on your node. Note that the VT100 definitions for the $\langle F2 \rangle$, $\langle F3 \rangle$, and $\langle F7 \rangle$ keys supercede the usual EMT definitions for these keys.

DOMAIN key	VT100 keypad
<ins mode=""></ins>	<esc></esc>
<char del=""></char>	<rubout></rubout>
<f2></f2>	<pf1></pf1>
<f3></f3>	<pf2></pf2>
<f4></f4>	<pf3></pf3>
<f5></f5>	<pf4></pf4>
SHIFT/ <f2></f2>	<7>
SHIFT/ <f3></f3>	<8>
SHIFT/ <f4></f4>	<9>
SHIFT/ <f5></f5>	<->
CTRL/ <f2></f2>	<4>
CTRL/ <f3></f3>	<5>
CTRL/ <f4></f4>	<6>
CTRL/ <f5></f5>	<,>
<f6></f6>	<1>
<f7></f7>	<2>
SHIFT/ <f6></f6>	<3>
SHIFT/ <f7></f7>	<enter></enter>
CTRL/ <f6></f6>	<0>
CTRL/ <f7></f7>	<.>

WBAK (WRITE_BACKUP) -- Create a magnetic media backup file.

FORMAT

WBAK pathname ... -F file_no [options] [-]

WBAK writes one or more objects to a magnetic media backup file. These objects may be directory trees, files, or links. For each object, the information saved includes the name, object data, and attributes associated with the object, such as the access control list. This lets you reconstruct files, the directory tree, or any portion of the tree using the RBAK (READ_BACKUP) command.

The WBAK and RBAK commands are intended both for disk backup and for interchanging information between separate DOMAIN installations. Use the RWMT (READ_WRITE_MAGTAPE) command to read and write magnetic media which are used for interchanging information with non-DOMAIN installations.

Tape Structure

WBAK writes the contents of the objects you specify to a single "backup file". Note that a backup file may be an ANSI standard tape file or diskette, and may contain many DOMAIN files, directories, and links. A backup file is a logically (and, if contained on one physical volume, physically) contiguous area of magnetic media surrounded by ANSI "file header" and "end of file" labels. One physical backup volume may contain one or more backup files. A single backup file may, however, span multiple magnetic media volumes.

The collection of backup files on one or more associated physical magnetic media volumes is called a "file set". The first backup file on the first physical magnetic media volume in a file set is numbered "1". Subsequent backup files in this file set are numbered in ascending order from "2".

Backup Modes

If you are backing up directory trees, WBAK can operate in one of three modes: "full" backup, "incremental" backup, or "dtm relative" backup. In full backup mode, all files, directories, and links are written to the backup file. In incremental backup mode, all files are saved which were modified since the last full or incremental backup (when the backup history file was updated). In dtm relative mode, all files which were most recently modified either before or after the specified time are written to the tape.

Backup History

WBAK records all times that a directory has been backed up in a file called BACKUP_HISTORY. This file is updated in all directories named on the command line with the 'pathname' argument; it is not updated in directories contained within (subordinate to) those named directories. If no directory is named on the command line, then no BACKUP_HISTORY file is made. The information written to this file includes the date and time of the backup (in Coordinate Universal Time (UTC)), the backup mode, and, if you have specified a dtm relative backup, the date and time to which the backup is relative. WBAK uses this file in incremental backup mode to determine the date and time of the

last full or incremental backup. This file is a standard text file and may be read in the same way as any text file; you should not, however, change it except possibly to delete old entries from the beginning of the file if it becomes too large. The automatic update of the history file can be suppressed by using the -NHIST command option.

File Identification on Tape

Associated with a backup file is a "file id" (-FID option). This is a 1 through 17 character user-assigned name which can be used in place of the file number to identify the backup file. This name is stored in the file header label and is printed (by default) by RBAK when the contents of a backup file are indexed (listed) or restored.

Full Disk Backup

An entire disk can be backed up by specifying the entry directory name as the pathname (example 2). It takes approximately 25 minutes to perform a full backup on a local 33 megabyte Winchester disk; 50 minutes for a remote disk.

Backup Verification

Use RBAK with the -INDEX option to verify a single backup file. For an index of all backup files on one physical tape volume, use RWMT with the -INDEX option.

When using WBAK, please note the following:

- Directories must allow list access in order to be backed up.
- Files must allow read access in order to be backed up.
- Objects locked for writing by another process cannot be backed up.
- WBAK must be run on the node which is connected to the tape or floppy unit. You may accomplish this either by physically typing the WBAK command on the host node, or by running WBAK in a process on the host node created from your own remote node using the CRP CREATE_PROCESS command.
- Only one tape unit can be connected to any node.
- There are no special tape mounting commands. Simply mount the tape on the tape drive and execute WBAK.

ARGUMENTS

pathname (required)

Specify the name of the object to be written to backup media. This may be a directory, file, or link. If it is a file, then the file is written as specified. If it is a link, then the link is resolved and the resolution object is written to backup media. If it is a directory, all subordinate files and subdirectories in the tree are

written. Note that the pathname specified reflects the way the directory is stored on the backup media, and that the same name must be used when reading files using pathnames in RBAK. Multiple pathnames and wildcarding are permitted. If you omit this argument, WBAK will prompt you for it. You may specify a hyphen (-) as an argument to direct WBAK to standard input for further arguments and options.

OPTIONS

The -F option is required, as it specifies where on the backup media the new file is to be written. If you omit it, WBAK will prompt you for it.

Default options are indicated by "(D)."

Tape File Identifiers

-FID file id

Specify a 1-17 character file ID to be written in the file header label for use when writing a file to a labeled volume. If this option is omitted, the name of the file being written is used.

-F [position]

Specify the file position for the write operation. Valid values for 'position' are "CUR", "END", or a nonzero integer. A position of "CUR" specifies that the file should be written at the current position on the backup media; the media must have been previously written by WBAK and its position must not have been disturbed.

A position of "END" specifies that the file should be written at the end of the backup media file set. This causes WBAK to append the specified disk file ('pathname' argument) to the very end of the file set.

A position specified by a nonzero integer value causes the file to be written at that absolute position in the backup media volume. If multiple 'pathname' arguments are supplied, the value of 'position' is incremented by one after each file has been written.

The default value for 'position' is 1.

Mode Control

-INCR

-AF dtm

The object specified by the 'pathname' argument must be a directory for either -FULL or -INCR to have meaning.

-FULL (D) Specify a full backup; save all files in specified trees.

Specify an incremental backup; save files which were modified since the last backup recorded in the BACKUP_HISTORY file stored in the 'pathname' directory.

Save all files modified after the given date and time; dtm is in the form "yy/mm/dd.hh:mm". The date defaults to today, and the time to midnight if either of those are omitted from dtm.

-BEF dtm Save all files last modified before the given date and time.

Label Control

-NWLA

Write the backup media volume label if the backup file number -WLA (D) is 1.

Suppress writing of the backup media volume label.

-OWN id Specify backup media volume owner (1-6 character name). This

option is only meaningful when used with the -WLA option.

Specify a 1-6 character volume ID for use when labeling a -VID vol id volume. This option is only meaningful when the backup file

number is 1. The default volume ID is '' (blank).

Display the label information written for this backup file on -SLA (D)

standard output.

-NSLA Suppress output of label information.

Listing Control

You may include the -L option, or any combination of -LD, -LF, AND -LL.

Write the names of all files, directories, and links saved to -L

standard output.

Write the names of all files saved to standard output. -LF

Write the names of all directories saved to standard output. -LD

Write the names of all links saved to standard output. -LL

Backup Device Control

Specify device type and unit number. 'd' must be either 'M' -DEV d[unit]

(for reel-to-reel magnetic tape), 'CT' (for cartridge tape), or 'F' (for floppy), depending on which drive is being used. 'unit' is an integer (0-3). Both are required for reel-to-reel tapes (i.e., -DEV M2). A unit number is NOT required for floppy disks and cartridge tapes (i.e., -DEV F). If this option is omitted, RBAK

assumes device M0.

Floppy disk support for this command is limited. CAUTION:

In particular, error detection during reads and writes is poor. DO NOT use this command with floppy disks when the data being placed on the

floppy disks are critical and unrecoverable.

Force previous backup media volume to be reopened, and -REO

suppress reading of backup media volume label. Use only when backup media has not been repositioned since last WBAK or

RBAK.

Special Cartridge Tape Control Options

-RETEN

Retension the cartridge tape (unwind to the end, then rewind). This can be helpful if you have encountered cartridge tape reading errors. Retensioning requires about 1.5 minutes to complete.

-NRETEN (D)

Do not retension the cartridge tape.

-NO_EOT

Suppress the writing of two tape marks at the end of the tape file, which are the standard signal for end of tape. The cartridge can't position between the two tapemarks to be ready for a successive call to WBAK (as it does on magtape), without rewinding the tape and searching forward, so this option speeds up multiple invocations of WBAK. It SHOULD NOT be used on the LAST invocation of WBAK. Also, '-F CUR' should be used on all WBAK invocations in a series except the first one.

-SYSBOOT

Permit use of a bootable tape that has a special boot program at the beginning. This option causes WBAK to skip over the first file on the tape. This option is only necessary when the first file on the tape is being written ('-F 1').

Miscellaneous Control Options

-NHI

Suppress update of the backup history file.

- (hyphen)

Read standard input for further arguments or options; input is accepted until WBAK receives an EOF (CTRL/Z by default).

This command uses the command line parser, and so also accepts the standard command options listed in the description of the command line parser in Chapter 3.

EXAMPLES

1. \$ wbak //mask/wby -f 1 -af 81/11/19.12.00 -fid wby -L

This command writes the directory //MASK/WBY to tape. The directory is written out to tape file one, and the file ID "wby" is written to the file's label. Disk files from directory WBY are written to the tape only if they have been modified since noon on November 19, 1981. The label and the names of the files written to tape are printed to standard output.

When this command is executed, WBAK writes the following information to standard output:

Label:

File number: 1
File section: 1
File ID: wby

Date written:

1981/11/20 10:47:58 EST

Starting write:

```
(file) "//mask/wby/among" written
(file) "//mask/wby/school" written
(file) "//mask/wby/children" written
(file) "//mask/wby/backup_history" written
(dir) "//mask/wby/" written.
```

Write complete.

2. \$ wbak -f 1 -own "john doe" -vid "volbk2" -fid "node 27 backup" //gooey

This command backs up the entire contents of the node whose entry directory name is "gooey". Note that the file ID is specified as "node 27 backup" to make it easy to identify when you want to reload it, and that the command assigns volume and owner IDs.

When this command is executed, WBAK writes the following information to standard output:

Label:

Volume ID: VOLBK2
Owner ID: john doe
File number: 1

File section: 1

File ID: n 27 backup

File written: 1983/02/17 18:00:39 EST

Starting write:

Write complete.

3. \$ wbak -f 1 -own "john doe" -vid "volbk1" ug/[a-f]?*_example -l

This command uses wildcards to match only those files in the "ug" subdirectory of the current working directory whose names begin with the letters "a" through "f" and end with " example".

When this command is executed, WBAK writes the following information to standard output:

Label:

Volume ID: VOLBK1 Owner ID: john doe

File number: 1
File section: 1

File ID: (no ID specified)

File written: 1983/02/17 17:58:52 EST

Starting write:

```
(file) "ug/cmf_example" written.
(file) "ug/cmt_example" written.
(file) "ug/cpboot_example" written.
(file) "ug/cpf_example" written.
(file) "ug/cpt_example" written.
(file) "ug/fpat_example" written.
```

WBAK (WRITE_BACKUP)

(file) "ug/fppmask_example" written.
(file) "ug/fst_example" written.

Write complete.

DIAGNOSTICS

I/O Errors

When WBAK has an I/O error, it attempts the operation again, for a total of five times. After the fourth retry fails, WBAK prints out an error message describing which type of error occurred. If the error was during an attempt to write to a tape, WBAK skips the tape block which caused the error, and tries to write the same data in the next block. Note that no data is lost, but RBAK will return an I/O error when it tries to read that block. If the write attempt again fails, after five tries, WBAK skips that block and tries the next. This process will continue for a total of twenty consecutive failed blocks, at which time WBAK aborts.

tape rewind error

An I/O error occurred.

tape write-filemark error

An I/O error occurred.

tape space-filemark error

An I/O error occurred.

tape space-record error

An I/O error occurred.

i/o recovery failed

An I/O error occurred and the tape drive could not reposition for another try.

tape i/o error

An I/O error occurred.

Operator Errors

first label on volume is not VOL1 label

Expected a standard label, and did not find one.

label version number in VOL1 label is not "3"

The label format is incorrect.

a HDR1 label is missing where one is required

A file on the tape does not begin with the correct format.

wrong volume, file header is inconsistent with previous trailer

The wrong continuation tape was put on the drive. This is an operator error which can occur when a multi-tape volume is used.

magtape drive is offline

You have not put the drive on line.

tape is write-protected

The write enable ring is not on the tape.

file not found

The tape file specified was not found.

invalid unit number

Tape unit specified is not connected. Presently, only DEV 0 is supported.

pbu is not present.

No tape unit is connected to the node. WBAK can only be run on the node connected to the tape drive.

WD (WORKING_DIRECTORY) -- Set or display the current working directory.

FORMAT

WD [pathname]

WD sets the working directory for the current process to the specified directory. The working directory is where the system looks for objects when you don't explicitly specify a directory as a part of a pathname.

ARGUMENTS

pathname (optional)

Specify new working directory. This may be a derived name, but must point to a directory or link to a directory. Specifying a file will cause an error. WD also accepts the command line parser arguments "-" and "*". If you specify a hyphen (-), WD looks to standard input for the directory name. An asterisk (*) followed by the name of a file directs WD to look inside that file for the new working directory name.

Default if omitted: display current working directory.

EXAMPLES

<pre>1. \$ wd //fred/jtj \$ wd //fred/jtj</pre>	Set new working directory. Display the new setting.
<pre>\$ wd stuff/revised \$ wd //fred/jtj/stuff/revise</pre>	Set working directory with derived name Display the new setting.
2. \$ wd - //frodo/my_stuff	Direct input to standard input for new directory name. Signal end of input with CTRL/Z.
3. \$ wd *newdir \$	Direct input to a file named "newdir" that holds the name of the new working directory.

WHILE -- Execute a WHILE loop.

FORMAT

WHILE condition DO command ... ENDDO

WHILE executes a command (or commands) as long as the results of a Boolean test are true. You can extend the WHILE command over several lines if you use it interactively or in a Shell script. When you use WHILE interactively, and extend the command over more than one line, the Shell prompts you for each new line of the command with the \$__ prompt.

ARGUMENTS

condition (required)

Specify a command or program to execute and test for truth, or specify a variable expression or Boolean variable to test for truth. "Truth" usually means that the command executes successfully (without any errors), or that a Shell variable expression or Boolean is "true". (Specifically, this argument is evaluated TRUE if it returns an abort severity level of 0 (zero).)

Refer to the DOMAIN System User's Guide for more information on Shell variables.

command ...

(required)

Specify the command(s) or program(s) to execute if 'condition' returns TRUE.

EXAMPLES

XDMC (EXECUTE_DM_COMMAND) -- Execute a DM command from the Shell.

FORMAT

XDMC dm_command [args...]

XDMC allows you to invoke Display Manager commands from the command Shell or from within a Shell script. This is similar to pressing the <CMD> key on the keyboard and then typing the DM command in the DM input window, which is the usual way to perform DM operations.

ARGUMENTS

dm_command

(required)

Specify the Display Manager command to be executed. See

Chapter 2 for DM command descriptions.

args ...

(optional)

Specify any arguments to be passed to the DM command. These are sent directly to the DM without further processing by the command Shell.

Default if omitted: no arguments passed

EXAMPLES

1. \$ xdmc dq

Cause the DM to send a quit fault to the current process. This is analagous to the SIGP (SIGNAL_PROCESS) Shell command, with the following important difference. Whereas SIGP accepts an argument designating which process to fault, this example will send a fault to whatever process (if any) is pointed to by the current cursor position.

2. \$ xdmc cp /com/sh

Cause the DM to create a new process and invoke the Shell. This is the same as pressing the <SHELL> key.

XOFF -- Deactivate the Shell's -X flag.

FORMAT

XOFF

XOFF turns off the Shell's -X (execution trace) flag, which is turned on by the XON command or by the -X option on the SH command. When the flag is off, command lines are not displayed before execution. The flag is off by default.

XOFF requires no arguments or options.

XON -- Activate the Shell's -X flag.

FORMAT

XON

XON turns on execution tracing. Just before each command is executed, its full pathname and arguments are written to the error output stream of the Shell. In Shell scripts, XON can be used to show the progress being made by the script, and can help debug Shell scripts by showing the actual arguments being passed to commands, after all Shell processing on them is complete.

By default, execution tracing is off when a Shell is invoked.

If XON is turned on in a Shell script, it remains on until that Shell script exits, or until over-ridden by a XOFF in a nested Shell script. When a Shell script exits, the state of execution tracing is returned to the state in effect just before the script was invoked.

XON requires no arguments or options.

XSUBS (EXECUTE_SUBSYSTEM) -- Run Shell script subsystem manager.

FORMAT

XSUBS pathname [args...]

Once a protected subsystem, a subsystem manager(s), and a subsystem data object(s) exist, any user can execute the manager program. To run a binary manager program, you simply execute the program. To run a Shell script manager program, you must use the XSUBS command. Note that in order to see the name of a subsystem created on another node, you must copy the file /SYS/SYBSYS/Subsystem_name to your node. If you do not copy this file, you can use the subsystem managers to operate on the objects, but when you ask to display the name of the subsystem, you will get an error message like the following:

ARGUMENTS

pathname

(required)

Specify Shell script containing the subsystem manager to be executed. Note that this script must contain the commands SUBS -UP and SUBS -DOWN in order to enter and exit the subsystem.

args ... (optional)

Specify arguments to be passed to the Shell script.

Default if omitted: no arguments passed

EXAMPLES

Suppose you have an append-only list that you wish to protect. Anyone can read the list, and append to the list, but no one can overwrite previously existing contents. Assume that the subsystem 'append_only' already exists. Then the 'APP' Shell script, which appends standard input to an append-only file, would look like this:

```
# APP --- append to an append_only file
SUBS -UP
CATF >>^1  # append to the file passed as first argument
SUBS -DOWN
```

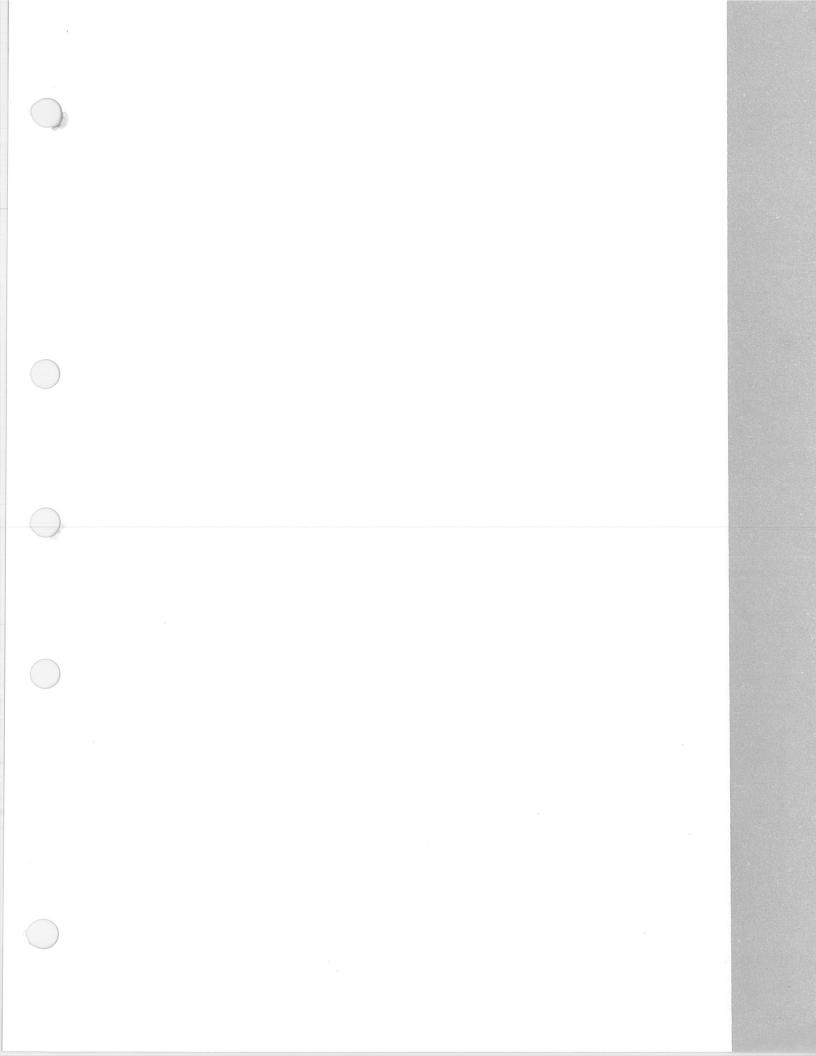
To make APP a manager of the 'append_only' subsystem, do

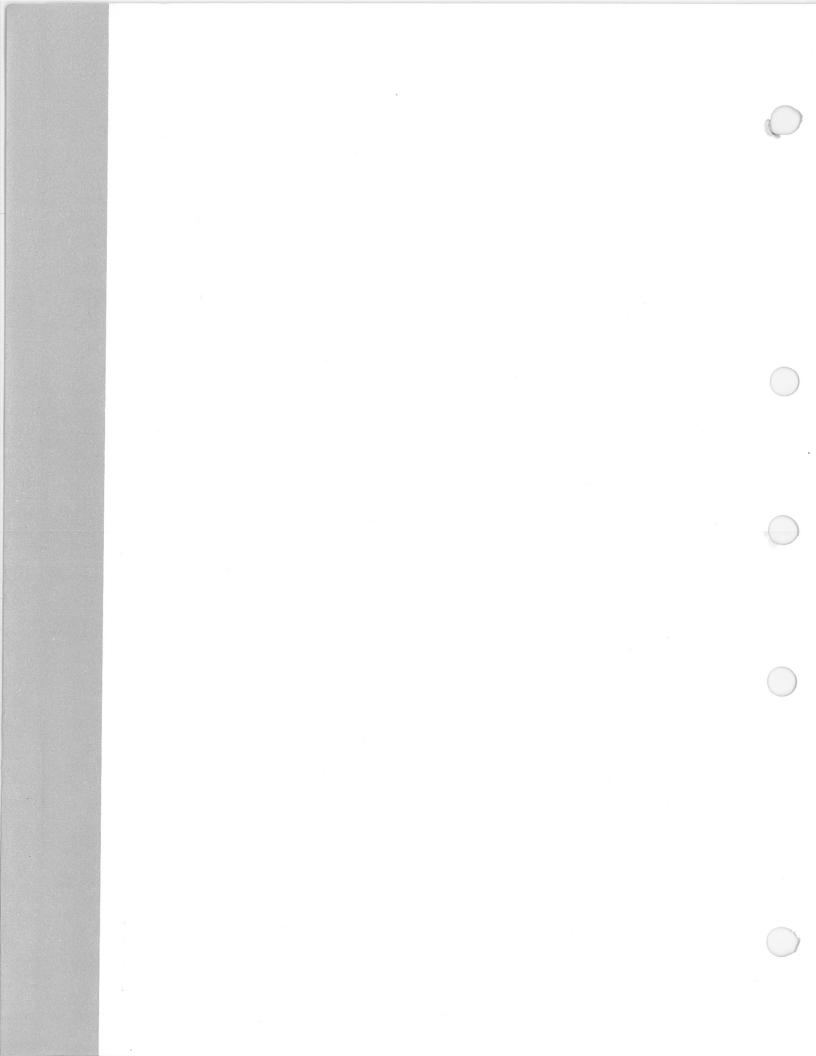
```
ENSUBS append_only # enter subsystem
SUBS APP append_only -MGR
*** EOF ****
```

XSUBS (EXECUTE_SUBSYSTEM)

A run of APP would look like this:

XSUBS APP aofile # execute APP on 'aofile' this is the stuff that is appended *** EOF ***





Appendix A CALENDAR

The calendar utility, CALENDAR, sets the date and time in the hardware calendar, and stores time zone information on a logical volume. You must use this utility at least once on any volume from which the operating system will be started. Its use is unnecessary on other volumes. CALENDAR must be run only on initialized disks.

You may run CALENDAR either from the Shell or from the Mnemonic Debugger. From the Shell, type

\$ calendar

and follow the prompts as described below.

To run the calendar utility from the Mnemonic Debugger, first shut down the Display Manger by typing SHUT in the DM input window. When you receive the Debugger prompt "> ", enter service mode by flipping the NORMAL/SERVICE switch on your node to the SERVICE position, then type the following command:

>EX CALENDAR <RETURN>

This command loads and starts the calendar as a stand-alone utility. The utility identifies itself, then requests the type of disk and the logical volume number. Type W for a Winchester disk, S for a storage module disk, or F for a floppy disk. To specify a unit number, append 0 or 1 to the letter. For example, S1 denotes storage module unit 1. Unit 0 is the default. Next, enter the number of the logical volume if it is not 1.

The utility now checks to see if the volume already contains any calendar information. If the calendar has already been set, the utility informs you of the settings and allows you to change them. If the calendar has never been set, the utility prompts you for the required information.

First, the utility requests a time zone. The time zone is necessary because all date/time information is recorded internally in Coordinate Universal Time (UTC). The time zone offset is the value which, when applied to a UTC time, produces a local time. You can specify either a time zone name (see below) or a positive or negative offset from UTC. If you specify an offset, the utility later prompts you for an optional 1 to 4 character identifier to be used as a time zone name. Offsets may be in whole or half-hour increments. Other fractional offsets produce an error message.

After you enter the time zone, the utility prints the date and time stored in the hardware calendar and asks if you want to make changes. If so, specify the date in the following format:

[yy]yy/mm/dd

Leading zeros are not required, and you can omit the century. For example, 84/6/7 specifies June 7, 1984.

Next, the utility requests the local time. Enter the local time in hh:mm format.

A warning message appears if you attempt to set the time backward, or attempt to set it forward

by more than five minutes. If you are sure that what you typed is correct, ignore the message. The utility then sets the date and time in the hardware calendar and exits.

When you have completed the CALENDAR routine, place the NORMAL/SERVICE switch back in its NORMAL setting.

Table A-1. Valid Time Zones

Name	Time Zone
EDT	Eastern Daylight Time
EST	Eastern Standard Time
CDT	Central Daylight Time
CST	Central Standard Time
MDT	Mountain Daylight Time
MST	Mountain Standard Time
PDT	Pacific Daylight Time
PST	Pacific Standard Time
GMT	Greenwich Mean Time
UTC	Coordinate Universal Time

Appendix B OLD_EDFONT

OLD_EDFONT is a function key-based, menu-driven program that allows you to interactively edit and view character font files. A newer version of the program (EDFONT) offers similar capabilities with an improved, pointing device-oriented user interface.

RESTRICTION

Due to the present font file format, character descriptor components (left, right, up, down, and width) must be within the range -127 to 127. The maximum size of a character image must not be greater than 224 pixels by 224 pixels.

B.1. Background

What is a Font?

A font is data that graphically describes a set of related character images. Fonts are stored in named files on a node. A font file contains exactly one font.

All of the characters which appear on a display are read from a font file. You can select the font to be used by using the Display Manager command FL (FONT_LOAD) or by calling PAD or GPR system routines from your own program. At least two fonts are always in use by a node: the standard font, contained in /SYS/DM/FONTS/STD (plus the ".19L" or ".COLOR" suffixes for the associated display types), and the window legend font, contained in /SYS/DM/FONTS/LEGEND (plus the ".19L" or ".COLOR" suffixes for the associated display types).

Fonts are generally classified into two categories: mono-spaced and proportionally-spaced. Characters in mono-spaced fonts are all exactly the same size, making these fonts most useful whenever column alignment is important. Character sizes in proportionally-spaced fonts vary from character to character. Proportionally-spaced fonts are generally used in typesetting and other document preparation applications.

Font Contents

The data in a font is broken into two parts. The first, called the **font header**, pertains to the entire font. This information includes the number of characters in the font and the maximum character sizes. Additionally, it contains the following three user-modifiable components:

horizontal spacing

the intracharacter spacing, in pixels. This is used to compute the horizontal position of each successive character written to the display.

vertical spacing the interline spacing, in pixels. This is used to compute the vertical position of each successive line.

space size the number of pixels to skip in the horizontal direction when a character is written which is not in the font.

The second part of the font describes each individual character. Two pieces of information are maintained for each character. The first is the **character image**. This is a two-dimensional bit-map which is moved to the visible portion of display memory to display the character. The second piece, called the **character descriptor**, describes how the character image is to be displayed.

Before going any further with the explanation of the character descriptor, it is necessary to understand how a string of characters is displayed by the system. The primitive that displays a string of characters, which is called SMD_\$WRITE_STRING, is supplied an (x,y) position at which to display the first character. This position is used as the "origin" for the first character of the string. It is not necessarily the lower left-hand corner of the character. When the first character has been written to the display, the horizontal component of the origin is updated, and the next character is written. This continues until all characters in the string have been displayed.

Each character descriptor has five scalar components. These are called the "up", "down", "right", "left", and "width" of the character. They have the following meanings:

up is the number of pixels above the origin used for the image, including the pixel

at the origin.

down is the number of pixels below the origin used for the image.

right is the number of pixels to the right of the origin used for the image, including

the pixel at the origin.

left is the number of pixels to the left of the origin used for the image.

width is one of the two values added to the horizontal component of the origin in

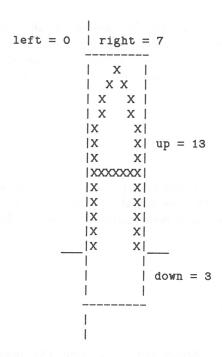
order to compute the origin for the subsequent character. (The intracharacter spacing is the other.) The width has no direct relationship with the image.

Each of these scalars is represented internally by a signed, 8-bit value. Negative values are allowed, however the magnitude of any value must not exceed 127.

The total number of pixels occupied by a character in the horizontal direction is computed by adding the left and right components. (OLD_EDFONT calls this the "total width".) Similarly, adding the up and down components produces the number of pixels occupied in the vertical direction (the "total height").

An Example

For example, let us consider the character "A" from the standard font for the 15 inch portrait display. The standard font is mono-spaced, with each character occupying a total of 7 pixels horizontally and 16 pixels vertically; 13 pixels are at and above the origin and 3 pixels are below the origin. (The pixels below the origin are sometimes called "descenders".) The character image looks like this:



To display this character, the system is given, among other information, its origin. Restated, the origin is the coordinate pair at which the pixel which has a "right" of 1 and an "up" of 1 is placed. For the "A", this corresponds to the lower- and left-most "X".

The extensions of the rectangle used in the diagram above let you quickly locate the origin. Simply find the intersection of the extensions. The pixel at the upper right of this intersection is the pixel at the origin. When OLD_EDFONT displays the character image grid, similar grid extensions are provided for locating the origin.

Suppose that our "A" was written with an origin of (5,80) (x,y). (The upper left-hand corner of the display is (0,0).) The image, therefore, will occupy columns 5 through 11, and rows 68 through 83. Although all of the pixels in rows 81 through 83 contain zero values, they are still occupied by the image.

B.2. Overview

OLD_EDFONT allows you to create new fonts and update existing fonts by selecting actions from a menu. The menu is tree-structured and easily traversed. The "pop" key, <F8> in all applicable menus, climbs the tree to the next higher level. A "define" menu which is available from most points in the hierarchy lets you define or redefine the parameters under which OLD_EDFONT is operating.

You interact with OLD_EDFONT through the keyboard, the touchpad, or the Summagraphics "Bit Pad One" tablet with a 4-button puck. Menu items are usually selected by either pressing the corresponding function key (i.e., press <F1> to select menu option 1, etc.), or by pointing to the desired menu box and pressing any of the buttons on the Bit Pad One puck. When editing a character image, the four buttons on the puck correspond to the first four function keys. The touchpad is only used for moving the cursor; it is not used for selecting menu items.

Character images are created and edited in a "grid" which is displayed in the center of the screen. Pixels are set and reset in the grid by pointing to the desired pixel and using the

appropriate function key or puck button. Lines may also be drawn in this manner. OLD_EDFONT maintains a copy of the character in its actual size, which is updated as you modify the grid. Also, a "sample text" capability is provided, where you may type characters which appear in the font you are editing. This allows you to see how different characters look when juxtaposed.

B.3. A Brief Walk-Through

Following is a brief example that allows you to create a new font (consisting of only a single character) based on alterations to an existing font. This should allow you to get a feel for OLD_EDFONT before we plunge into the details of every menu option. Please perform the operations described as you read through the example.

1. Invoke OLD_EDFONT by typing

\$ old_edfont <RETURN>

in the Shell's process input pad.

2. OLD_EDFONT displays the top-level menu. Select option 3 from the menu displayed by pressing the F3 key.

١	1	Ī	2	1 3	Ī	4	ī	5	Ī	6	Ī	71	8	-
- 1	open	1	open	create	1	define	1		ĺ		1	la constal o	quit	1
1:	for read	for	r update	lnew font	1				1			100		1
1		_		1	_1	t contract	_1		-	11 7 T	1	I		_

3. Type a pathname in reponse to the "Font file:" prompt.

Font file: my_new_font

OLD_EDFONT creates the empty font file and displays the main EDIT menu.

4. Select option 5 to display the DEFINE menu, where we will specify the name of the alternate font which we will use as the basis for the characters in the new font.

1	1	T	2	1	3 4	ī	5	1	6	1 7	ī	8	-1
i	edit	i	edit	į	delete display	i	define	i		close	i	close	i
	header	cl	naracter	1	character map	1		1		no update	١u	pdate	1
1		1_		1		١		_1		1	1_	1000	1
							^	_					-

5. Select option 3 to declare the pathname of the alternate font. In response to the "Alternate font file:" prompt, type "/sys/dm/fonts/old_english.36".

Alternate font file: /sys/dm/fonts/old_english.36

1	2	3	4	5	6	7	8
pointing	character alte	rnate def	ault tou	chpad	1	sample	l pop l
device	defaults fo	ont gr	id m	ode	1	text	
		I	1	<u> </u>	1		
		^					

OLD_EDFONT will respond "Opened." when the alternate font file has been opened. Press option 8 to leave the DEFINE menu. OLD_EDFONT returns to the main EDIT menu.

Select option 2 to edit a character. Type "A" when OLD EDFONT asks which
character you wish to edit.

Character: A

1	1	1 2	3	1	4	1	5	ī	6 7 8
1	edit	edit	delete	d	isplay	1	define	1	close close
	header	character	characte	rl	map	1		1	no update update
	1.5	1		1	3	1		1	
		^							

OLD_EDFONT responds "Create 'A' (41)" in the upper right corner of the screen and then displays the EDIT CHARACTER menu.

7. Select option 5 to copy a character from the alternate font. Type "A-a" in response to the "Character:" prompt. (The -a option indicates that you want the character copied from the alternate font. This is important, since there aren't any characters in the current font yet.)

Character: A -a

_															
1	1		2	I	3	T	4	5	1	6		7	T	8	
	edit		edit	1	next	1	another	сору		define			1	pop	1
d	escriptor		image	10	haracter	1	character	character	-				1		
1		1		1		1	1		1		1		1		
-		_		-					-				_		

OLD_EDFONT responds "Copied." and writes the copied character into the sample text window outlined at the lower right of the screen so that you can make sure the correct one was copied. The menu does not change.

8. Select option 2 to edit the image of the new character (which currently mimics the Gothic capital "A" from the system's Old English font). OLD_EDFONT displays the character image on the default grid and displays the EDIT IMAGE menu.

1	1		2	I	3	4	T	5	1		6	1	7	T	8	-
1	edit	edit		nex	t	another	1	сору	1	defin	е				pop	1
de	scriptor	imag	е	chara	cter	character	10	haracter	-1					1		1
1				1			1_		1_				 	1_		_1

9. You may now edit the image by turning the various pixel squares on and off. You move the crosshairs which mark the current pixel by pressing the arrow keys at the left of the keyboard. Experiment with the functions of the various options (see the description of the EDIT IMAGE menus later in this appendix if you don't feel particularly adventurous). There are so many options, in fact, that it takes two menus to hold them. Switch menus with option 6 ("next"). When you are finished making your changes, select option 8 to stop editing and update the new character image.

1	1	I	2	T	3	-	4	Ī	5	Ī	6			7	T	8	1
	invert	1	mark		fill	1	fill	1	define	1	next		рор		1	pop	1
I	pixel	1	pixel	W	ith 1's	l li	ith 0's			1	1	no	upda	te	lu	pdate	1
1		. _		. _		1_		1		1					1_		1

10. Select option 8 to leave the EDIT CHARACTER menu (unless you want to return to step 7 and do another character.)

1 1	2	1 3	1 4	1 5	6	7	8
edit	edit	next	another	сору	define	1	pop
descriptor	image	character	clcharacter	character		1	
	1		_I	.li	l	1	1
							^

11. Select option 8 to update the new font file (containing one character at this point) on disk and leave the main EDIT menu.

	1	1	2	1 3	1	4	Ī	5	Ī	6	7	8	1
1				delete			1	define	1	1	close	close	ŀ
	header	characte	er	character	1	map	-		1	lno	update	update	1
1		· · · · · · · · · · · · · · · · · · ·					1		1	I		•	İ

OLD_EDFONT will respond "Font updated." and return to the top-level menu.

12. Select option 8 to leave OLD_EDFONT and return to the command Shell.

1	2	3	I	4	1	5	6	1 7	8	-
	open open				1		1	1	quit	1
for read	for update	new font			1		1	I	1	1
		1	_1		1_			l	l	_1
									-	

B.4. Detailed Menu Descriptions

Now that you've completed a simple example, let's examine all of the options and capabilities that OLD_EDFONT provides. To assist you in placing each menu within the hierarchy, you will find a menu identifier in the right-hand margin of the descriptions below.

To invoke OLD_EDFONT, type the Shell command

\$ old edfont <RETURN>

OLD_EDFONT displays the top-level menu at the bottom of the display.

1	2	3	4	1	5	6	71	8
-	open c			1	1	İ	i	quit
for read	for update ne	w font		1		1		1
						1		1

The OLD_EDFONT revision number is displayed at the upper left corner of the display. In the upper right corner is the crosshair, which you will use later when creating and editing character images.

Use option 1, "open for read," to view an existing font file. This option may be used whether or not the font is currently in use.

Use option 2, "open for update," to update an existing font file. Option 2 may be used only if the desired font file is not in use.

Use option 3, "create new font," to create a new font file.

After selecting any of these three options, OLD_EDFONT prompts you for the name of the font file that you wish to use. Provide the pathname.

Option 4, "define", causes the DEFINE menu to be invoked; this is described below.

Option 8 causes OLD_EDFONT to quit and return to the program which invoked it, usually the command Shell.

The DEFINE Menu

DEF

The DEFINE menu is available at several levels in the menu structure of OLD_EDFONT, so we discuss it first. It is used to view and set miscellaneous parameters and defaults used by OLD_EDFONT. These include the pointing device, grid size, character defaults, etc. The define menu looks like this:

1	2	3	4	5	6	1	7	T	8	l
	character							1	pop	
device	defaults	font	grid	mode	grid	1	text			
						_		1_		

Depending upon the place in the hierarchy from which define mode was entered, some of these menu items may not be displayed. For example, if no font file is open, then defining sample text is not meaningful. Also, if no grid is displayed, then defining the current grid is not meaningful.

To return to the menu from which you entered define mode, use <F8>.

All menus which are subordinate to the define menu pop directly to the menu which invoked define mode; they do not return to the define menu.

The POINTING DEVICE Menu

DEF/POINT

The POINTING DEVICE menu is entered from option 1 in the DEFINE menu.

1 2 1	3	I	4	I	5	6	1	7	8
keyboard & Bit Pad 1				1		1	1	1	pop
touchpad tablet				1		1	1	1	1
				1		1	1		

Select option 1 to use the keyboard and touchpad (the default), or option 2 to use the Bit Pad One tablet. Select option 8 to exit this menu.

If you select option 2, the menu is modified to appear as follows:

	1	2	3	4	5	6	7	I 8 I
keyboa	ard & Bit	Pad 1	i	l si	io 1	sio 2	sio 3	l pop l
	npad tal			and the same of		i		i
	1	1	-	ĺ		i		1

Select the menu item which corresponds to the SIO line to which the Bit Pad One is attached. OLD_EDFONT is configured to use the Bit Pad One at 9600 baud only.

From this point on, use the tablet to select menu options by placing the crosshair in the desired menu box and pressing any button. The function keys may also be used, providing that the puck is resting on the tablet when a function key is depressed. To disable the tablet, re-enter this menu and select menu option 1.

The tablet should not be enabled in this manner if the "sbp1" program has been run. (This program causes the Bit Pad One to act as though it were a touchpad.)

The CHARACTER DEFAULTS Menu

DEF/CHARDEF

Use the CHARACTER DEFAULTS menu (displayed by selecting option 2 in the DEFINE menu) to define the character descriptor which will be used for newly created characters. Selecting this menu causes the character defaults currently in effect to be displayed, as well as the new menu. The display will look like this:

Default character descriptor:

up: 13
down 3
left: 0
right: 8
width: 8

(total height: 16) (total width: 8)

auto width adjustment: off

1	1	I	2	3	Ī	4	T	5	6	 7	7	T			8	1
1	up	1	down	left		right	1	width auto	width				ро	p		1
1		1	64 61.00				1	ad	just							
		1					1		1			1				

Use menu options 1 through 5 to change the corresponding value in the default character descriptor.

Normally, the width for a character is given explicitly, however OLD_EDFONT can automatically compute the width. To compute the width, select menu option 6 and specify an offset which, when added to the "right" component of the character, produces the width. This value may be positive, zero, or negative. To disable auto width adjustment, select option 6 and enter "off".

If auto width adjustment is enabled, it is applied to existing characters if they are edited. This option is off by default.

When you have completed redefining the default character descriptor, select menu option 8.

Defining the Alternate Font

DEF/ALTFONT

Selecting option 3 in the DEFINE menu allows you to specify an alternate font file. This file is used only as a source for copying characters. The "copy character" option is described later.

Use of this option first causes any current alternate font file to be closed. You are then prompted for the name of a font file.

The DEFAULT GRID Menu

DEF/DEFGRID

The DEFAULT GRID menu (display by selecting option 4 in the DEFINE menu) causes the current grid parameters to be displayed and allows you to change them. The display will look like this:

Default grid descriptor:

horizontal pixels: 8
vertical pixels: 16
pixel size: 22
horizontal offset: 0
vertical offset: 0
auto grid adjust: on

-												
	1	2	3	1	4	5	1	6	7	1	8	
hor	rizontal ve	rtical	pixel	h	orizontal	vertical	auto	grid		1	pop	
F	oixels p	ixels	size	1	offset	offset	adj	ust		1		
1				1				1		1		١

The default grid descriptor specifies the parameters for the grid used for creating and editing character images. Select the option corresponding to the parameter whose value you wish to change. OLD_EDFONT will then prompt you for the new value. These parameters are described below.

horizontal pixels is the number of pixels represented by the grid in the horizontal direction.

vertical pixels is the number of pixels represented by the grid in the vertical direction.

pixel size is the size of each pixel "box" within the grid. The smaller this number, the larger the values for horizontal and vertical pixels may be.

horizontal offset is the offset, in pixels from the left side of the image, at which to start displaying the image within the grid. Positive values effectively shift the image to the left within the grid; negative values shift the image to the right. This is useful for editing a portion of an image if the entire image won't fit within the grid.

vertical offset is the offset, in pixels from the top edge of the image, at which to start displaying the image within the grid. Positive values effectively shift the image up within the grid; negative values shift the image down.

auto grid adjust if enabled, causes the grid to be adjusted to fit the image being displayed. If disabled, the five parameters above are used to describe the grid.

Modifying the default grid descriptor has no effect on a currently displayed grid. To modify the current grid use the "define current grid" option.

The TOUCHPAD MODE Menu

DEF/TOUCHPAD

The TOUCHPAD MODE menu (displayed by selecting option 5 in the DEFINE menu) allows you to redefine the mode of the touchpad, without exiting to the command Shell to use the TPM (TOUCHPAD_MODE) command. When this menu is selected, the current touchpad mode and the new menu options are written to the display. The display will look like this:

Touchpad mode:

mode:

relative

x scale:

200

y scale: hysteresis: 300

1	2	3	4	1	5	6	7	1	8
absolute	relative	absolute/ x	scale	lу	scale	hysteresis		1	pop
		relative		1		1		1	
l		11_							

Menu options 1 through 3 define the mode in which the touchpad operates. Menu options 4 and 5 define the scales used in the x and y axes, while menu option 6 defines the hysteresis. See the description of the TPM command for a complete description of these parameters.

The CURRENT GRID Menu

DEF/CURGRID

The CURRENT GRID menu (displayed by selecting option 6 in the DEFINE menu) is similar to the DEFAULT GRID menu, except that the parameters apply only to the grid which is currently displayed. The default grid descriptor is used for all future grids.

The SAMPLE TEXT Menu

DEF/TEXT

The SAMPLE TEXT menu (display by selecting option 7 in the DEFINE menu) allows you to specify the text which is to be displayed in the sample text box. The text box is the rectangle displayed to the lower right of the grid. When this menu is selected, a blinking cursor appears in the sample text box and the menu is displayed as follows:

1	1	2	3	4	5	6	7 8
	clear show a	11	1	1	1		pop
	text charact	ers		1	1		1 1
	ter 100 Principles		1 1500	T			

To clear all characters in the sample text box, depress <F1>. To display all of the characters that are in the font, depress <F2>. If all of the characters don't fit, as many as do fit are displayed.

You may also type characters on the keyboard to view them in the sample text box. The only special keys that are available are backspace (BS) and line delete (L2). The contents of the box will scroll when the box is full.

Note that the Bit Pad One tablet may not be used while in this menu; you must use the keyboard.

That completes the description of the DEFINE menu. We now return to the top-level menu displayed when OLD_EDFONT is invoked from the Shell.

The EDIT menu is the second-level menu which is used after a font file has been opened. This menu is entered from the top-level menu, described earlier. Since the menus used to view or edit a font are very similar, they are discussed together in the following sections.

This menu is displayed if the font file was opened for editing or was just created (options 2 and 3 in the top-level menu):

1	1	2	3	1	4	T	5	T		6	T	7	8
1	edit		delete			1	define	1			İ	close clos	se
-	header	character	characte	rl	map	1					n	update updat	te
1		11 15 15 15 15 15 15 15 15 15 15 15 15 1		_ _	9/17/5/17	1	a ni ahon	1_	2 39		1_	Aspenda I Noos	1

This menu is displayed if the font file was opened for reading (option 1 in the top-level menu):

1	2	1 3	4	5		6	7 8
view header	view character		display map	define	İ		close
					1		

In addition, the name of the font file replaces the revision number of OLD_EDFONT in the upper left corner of the display.

Option 1 allows you to edit or view the font file header. This produces a new menu, which is described below.

Option 2 allows you to edit or view a character image and/or descriptor. When option 2 is selected, OLD_EDFONT prompts for the character to be edited. Type either the ASCII character itself or its 2-digit hexadecimal code. Valid codes are in the range 00 through 7F. OLD_EDFONT then displays the character (and its code) in the upper right corner of the screen. This produces a new menu, which is described below.

If the font has been opened for editing, option 3 allows you to delete a character. When this option is selected, OLD_EDFONT prompts you for the character to delete. Enter either the character to be deleted, or its 2-digit hexadecimal code. Valid codes are in the range 00 through 7F.

Option 4 displays a map of the font. Both the hexadecimal code and the ASCII equivalent for each character are displayed. If the character is present in the font, an asterisk follows the code.

Option 5 enters the DEFINE menu, described earlier.

If the font has been created or opened for editing, option 7 closes the font file without performing any of the updates that were made during this OLD_EDFONT session.

Option 8 closes and updates the font file. If the font has been opened for reading, option 8 only closes the font file without updating it.

The EDIT HEADER and VIEW HEADER menus (displayed by selecting option 1 in the main EDIT and READ menus, respectively) are used to update or display the font header. When this menu is selected, the current font header is displayed, along with a new menu:

Font file header:

characters in font:	105	{total number of characters defined}
maximum height:	16	{maximum (up+down)}
maximum width:	8	{maximum (left+right)}
horizontal spacing:	1	{intra-character spacing}
vertical spacing:	3	{intra-line spacing}
space size:	8	{space to leave for undefined chars}

1 1	2	T	3	Ī	4	T	5	T	6	1	7	T	8
horizontal	vertical		space			1		1		1		1	pop
spacing	spacing	1	size			1		1		1		1	
11	197	1_		1		1		1		1		1	1

For the VIEW HEADER menu, the only option is "pop".

To modify the horizontal spacing, vertical spacing, or space size, select the appropriate option. OLD_EDFONT will prompt for a new value.

To return to the main EDIT menu, select option 8.

The EDIT CHARACTER and VIEW CHARACTER Menus

INV/EDIT/CHAR

The EDIT CHARACTER and VIEW CHARACTER menus (displayed by selecting option 2 in the main EDIT and READ menus, respectively) are used to display, create, and update a character in the font.

The menu looks like this when the font has been created or opened for editing:

		-												 			
	1		2	2		3	1	4		5		6	1	7	1	8	1
1	edit		edit		next			another	1	сору	1	define	1		1	pop	1
d	escriptor		image		charac	ter	-	character	10	haracter	-				1		1
1_		_	a 1 8 7 700		1				1		-		1		1		-

If the font has been opened for reading, the menu looks like this:

1 1	2	1	2	4	5	6 I	7 1	0 1
1 + 1	4	1	0	- I	9 1	0 1	· 1	0
view	view	nex	t and	other	de	fine	1	pop
descriptor	image	chara	cter char	racter	1	1	1	1
			1		1			

Option 1 allows you to edit or view the descriptor for this character. These options invoke a new menu which is described later.

Option 2 allows you to edit or view the image for this character, displayed using the default grid. These options are also described later.

Option 3 allows you to edit or view the character with the code which immediately follows the current character. (If "a" is being edited, "next character" would allow you to edit "b".)

Option 4 allows you to select another character to be edited or viewed.

If the font has been created or opened for update, option 5 allows you to copy the image of an existing character into the image of the character being edited. OLD_EDFONT prompts for the character to be copied: type either the character itself or its hexadecimal code. If you wish the character to be copied from the alternate font, type "-a" after the character, separated from it by a blank. (See step 7 in "A Brief Walk-Through" above.)

Option 6 causes the DEFINE menu to be invoked. This menu is described earlier.

Option 8 returns you to the main EDIT menu.

The EDIT DESCRIPTOR and VIEW DESCRIPTOR Menus

INV/EDIT/CHAR/DESC

Use the EDIT DESCRIPTOR and VIEW DESCRIPTOR menus to view or update the descriptor for a character. The current character descriptor is displayed along with the new menu. The display looks like this:

Character descriptor:

up: 13
down 3
left: 0
right: 8
width: 8

(total height: 16) (total width: 8)

auto width adjustment: off

1	1	2	T	3	I	4	T	5	1	6	7		8	1
up	1	down	1	left		right	1	width	lau	to width	pop	pop		1
	1						1		1	adjust no	update	update	3	
	_ _		1_		1		1		1					1

If the font is open for reading, the character descriptor is displayed, however the menu only offers the "pop" option.

The description of the CHARACTER DEFAULTS menu, labeled DEF/CHARDEF and found earlier in this section, applies to this menu as well. The only difference is the addition of option 7, which causes any updates made in this menu to be ignored. If option 8 is used, the changes made while in this menu are applied to the font.

The EDIT IMAGE and VIEW IMAGE Menus, Part 1

INV/EDIT/CHAR/IMAGE1

The EDIT IMAGE and VIEW IMAGE menus are used to create, update, and view a character image. When one of these menus is selected, the grid and actual-size character are displayed. The complete menu has 16 items, and therefore is split into two parts. The first part of the menu looks like this when creating or updating a font:

ı	1	Ī	2	3	Ī	4	T	5	Ī	6	7	8	
1	invert	1	mark	fill		fill	1	define		next	pop	pop	
1	pixel	1	pixel	with 1's	V	with 0's				no	update	update	
1	_	1			1		1						

If the font is open for reading, the menu looks like this:

										and the second second		
1	1	1 2	1 3	1	4	I	5	T	6	7		8
1				1		1	define	1				pop
-			1					1			1	
1						1		-		 	1_	

If the keyboard is being used as the "pointing device", a crosshair appears within the grid. The crosshair is controlled by the keys on the left-hand keypad (L1 through L15), as follows:

_				
		 top line 		
	left edge	 bottom line	right edge	 left edge
	up and left	 up 	up and right	up and
	left	.	right	 left
	down and left	 down 	down and right	down and

Low-	prof	ile	Keyboard

880 Keyboard

down

top line

bottom

line

up

right

up and

right

right

down and right

edge

Keyboard Map Codes

L2 top line

L4 left edge L5 bottom line

right edge L6

L7 up and left

L8 up

L9 up and right

L10 left

L12 right

L13 down and left

L14 down

L15 down and right

The four buttons on the Bit Pad One puck correspond to the first four function keys when the crosshair is positioned within the grid. The buttons are numbered as follows:

F1

F4 F3

F2

As you move the crosshair within the grid the current grid position is displayed on the top line of the display. The coordinates are displayed in terms of the position relative to the origin of the character. The first value is the horizontal position, which is preceded by either "L", when left of the origin, or "R" when right of the origin. The second is the y coordinate, preceded by either "U" when above the origin (up), or "D" when below it (down).

Option 1, "invert pixel", inverts the pixel under the crosshair. This means that, if the pixel is "on" (bright), this option will turn it "off" (dark). Likewise, if the pixel is off, this option will turn it on.

Option 2, "mark pixel", marks the pixel under the crosshair. This option is used when drawing and clearing lines (options 3 and 4). Marked pixels are identified by a half-tone in the grid position. Marking a pixel does not affect its value (on or off). The mark may be cleared by positioning the crosshair to the pixel and selecting the mark option again.

Option 3, "fill with 1's", draws a line from the pixel which was marked with option 2 to the pixel under the crosshair. Arbitrary slope lines may be drawn. If no pixel has been marked, the pixel under the crosshair is turned on.

Option 4, "fill with 0's", clears a line in the same manner that option 3 draws a line.

Option 5, "define", enters the define menu, described earlier.

Option 6, "next", selects the other half of the edit image menu. This is described below.

Option 7, "pop no update", causes any changes made while in the edit image menu to be ignored. OLD_EDFONT returns to the edit character menu.

Option 8, "pop update", causes any changes made in this menu to be applied to the font. OLD_EDFONT then returns to the edit character menu.

The EDIT IMAGE Menu, Part 2

INV/EDIT/CHAR/IMAGE2

The second part of the edit image menu contains less frequently used edit options. The menu looks like this:

1	1	0/3		2			3	1	4	T		5	T	6	T	7	8	-1
	set	1	mark			inser	t		delete	1	set		1	next	1	clear	invert	1
1	origin										limit	S	1			image	image	1
	ol waren	_ _	14 187 6	lds:	100			1	go huil	1	10912	97	1	1940 se	1_	endar in		1

This portion of the menu is available only when creating or updating a font. None of the options appear when reading a font.

Option 1, "set origin", sets the origin of the character to the pixel under the crosshair. The origin is the pixel location which is "1 up" and "1 right". The origin markers on the grid are updated to reflect the new origin.

Option 2, "mark", is not currently used in this menu.

Option 3, "insert", invokes the INSERT menu, described below. This allows you to insert pixels in the grid without modifying each affected pixel manually.

Option 4, "delete", invokes the DELETE menu, described below. Similar to insert, this allows you to delete pixels from the grid easily.

Option 5, "set limits", invokes the SET LIMITS menu. This option allows you to specify the limits of the character image graphically. (It is similar in function to the edit descriptor menu,

which requires you to enter numeric values for the image limits.) The SET LIMITS menu is described later.

Option 6, "next", returns to the first part of the EDIT IMAGE menu.

Option 7, "clear image", erases all pixels in the image. When you use this option, OLD_EDFONT queries you to ensure that you haven't made a mistake.

Option 8, "invert image", inverts all pixels in the image.

The INSERT Menu

INV/EDIT/CHAR/IMAGE2/INS

The INSERT menu is invoked from the second part of the EDIT IMAGE menu. It allows you to insert pixels into the grid by shifting the pixel data either vertically or horizontally. Also, complete rows and columns may be inserted into the grid. The menu looks like this:

-		-					17	and the second		<u>a da batan makina melabaki a</u>		
	1	1	2	3		4	T	5	1	6	7	0 1
		:	- !	_		-	•	0	1	0 1	, ,	0
	insert	1	insert	insert		insert	1	delete	1	1		pop
12									:	113 1 13 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1		Pop
l U	orizonta.	L۱۱	vertical	1 row	1:	l column	1					1
1		1	1		1		i		10			
1_		_!_					1		1			1

Option 1, "insert horizontal", inserts a pixel at the grid position under the crosshair and shifts pixels on its right to the right.

Option 2, "insert vertical", inserts a pixel at the grid position under the crosshair and shifts pixels below it down.

Option 3, "insert 1 row", inserts a row of pixels at the row under the crosshair and shifts all rows below it down.

Option 4, "insert 1 column" inserts a column of pixels at the column under the crosshair and shifts all columns on its right to the right.

Option 5, "delete", enters the DELETE menu. This option is available in this menu for convenience. (It is the same delete option as is in the edit image menu.)

Option 8, "pop" returns to the EDIT IMAGE menu.

The DELETE Menu

INV/EDIT/CHAR/IMAGE2/DEL

The DELETE menu is invoked from the second part of the EDIT IMAGE menu. It allows you to delete pixels from the grid by shifting the pixel data either vertically or horizontally. Also, complete rows and columns may be deleted. The menu looks like this:

1 1 2 1	3	4	5	T	6	7	8
delete delete	delete	delete	insert		1	pop	1
horizontal vertical	1 row	1 column		T	1		1
lI		.		_ _	1_	1	1

Option 1, "delete horizontal", deletes the pixel at the grid position under the crosshair and shifts pixels on its right to the left.

Option 2, "delete vertical", deletes the pixel at the grid position under the crosshair and shifts pixels below it up.

Option 3, "delete 1 row", deletes the row of pixels under the crosshair and shifts all rows below it up.

Option 4, "delete 1 column" deletes the column of pixels under the crosshair and shifts all columns on its right to the left.

Option 5, "insert", enters the INSERT menu. This option is available in this menu for convenience. (It is the same insert option as is in the EDIT IMAGE menu.)

Option 8, "pop" returns to the EDIT IMAGE menu.

The SET LIMITS Menu

INV/EDIT/CHAR/IMAGE2/LIM

The SET LIMITS menu allows you to define the limits of the character image, thereby setting the contents of the character descriptor. The SET LIMITS menu looks like this:

1		1	2	T	3		4	5	1	6	7	1	8	1
1	up		down	1	left	right	,						pop	
1				1							1			
											1	_	•	

To set the upper limit of the image, place the crosshair on the top-most row that you wish to save and select option one. When you do this, all rows above the selected upper limit are deleted from the image. Perform similar actions for the bottom row (use option 2), the left-most column (option 3), and the right-most column (option 4).

If you are using the tablet as the pointing device, you will notice that the keys on the puck correspond to the limits that they set. (The right key sets the right limit, for example.)

Appendix C EDFONT

Editing a Character Font

C.1. Introduction

EDFONT is a menu-driven program that allows you to design your own letters, numbers and/or special characters by constructing them, one at a time, on a screen grid. This grid is displayed as part of the initial EDFONT screen image. You can designate any character(s) you have created as a new font, store this font in a file and access it later for editing or printing. EDFONT allows you to do the following:

- Design a character by using grid spaces (pixels), lines, arcs, and filled or outlined circles and boxes
- Change the space size between characters
- Change the space size between lines
- Change the grid/character size
- Rotate a character on the grid
- View an actual size replica of the character you are currently designing
- Replace the character you are currently editing with one from the same or an alternate font file

In addition to designing your own fonts, you can modify existing system fonts. Refer to the FL (FONT_LOAD) command description in Chapter 2 for further information.

Following is a procedural description of the EDFONT program. See the Glossary of Terms at the end of this appendix for defintions of unfamiliar terms. The Glossary is also useful for background information, as is the "Background" section of the OLD_EDFONT description (Appendix B).

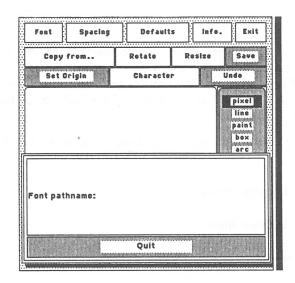
C-1

C.2. Invoking EDFONT

This program is designed to be used with either a mouse or a touchpad. To invoke EDFONT, type the following in the Shell input pad.

EDFONT <RETURN>

The initial screen image is displayed.



The font pathname specifies the name of a font file. It can be either a new font name or the name of an existing font. Enter the font name in the following form:

//_____<RETURN>

You can also invoke EDFONT by entering the name of the font right on the Shell input line. Example: EDFONT /SYS/DM/FONTS/STD

To exit, simply position the cursor on Exit and press either the Function 1 key (F1) or the leftmost Mouse key (M1). See item 5, following, for more information on Exit.

NOTE: The F1 and M1 keys are hereafter referred to collectively as the Select key.

C.3. Sample EDFONT Display

Figure C-1 shows a sample screen display. The following numbered items correspond to the numbered items in Figure C-1.

NOTE: You can get additional information about any item on the display by pressing the SHIFT and HELP keys at the cursor position where you need help. Move the cursor out of the help box to return to the original display.

- 1. Font
 When you position the cursor here and press the Select key, EDFONT displays the current font pathname. (For a description of how to enter the font pathname, see INVOKING EDFONT.) To keep the same name, simply move the cursor out of the pop-up. To change the name, use the LINE DEL key to delete, then enter the new font name and press RETURN. Position the cursor on Load This Font and press the Select key. If you have modified the font, EDFONT asks if you want to discard those modifications. Position the cursor on the appropriate box and press the Select key.
- 2. Spacing Select this option to change font values for vertical spacing, horizontal spacing, or space size. (Refer to the Glossary of Terms for definitions.) Position the cursor at the appropriate line, enter the new information and press the RETURN key.

NOTE: These values pertain to the whole font, not only to the character being edited.

3. Defaults

This option allows you to change or create default values for character width and/or height. (Refer to the Glossary of Terms for definitions.) These are the values that EDFONT will assign to the width/height of any newly created character.

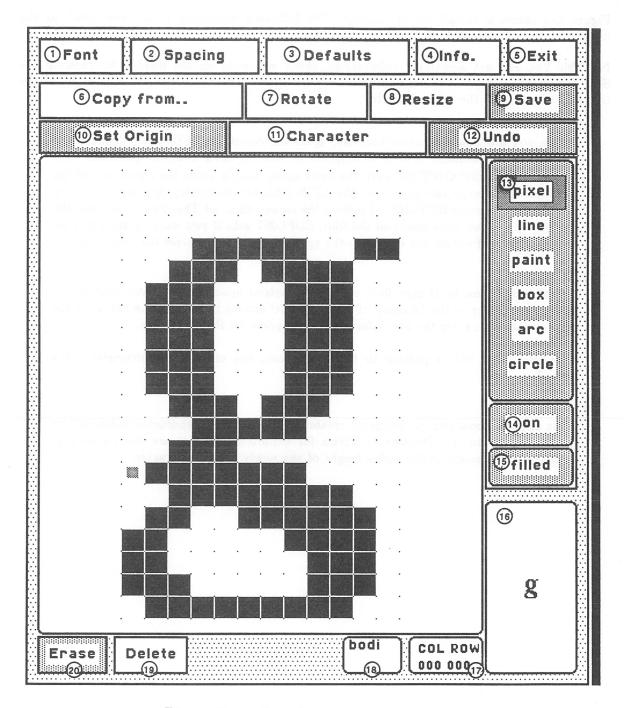


Figure C-1. Sample EDFONT Display

4. Info

To view information about the current font (number of characters in font, maximum height and maximum width) press the Select key at this option. You cannot modify these values.

5. Exit

Select this option to quit. If you have not modified the font, then EDFONT will quit immediately. If you have made modifications, then EDFONT will ask if you want to keep or discard them. If you decide to keep the modifications, the font file will be updated with the new design information.

6. Copy From

When you select this option, you can change the character image on the grid to another character in the same or in a different font. Enter the name of the font in which you wish to view this character and press RETURN. Next, type the character you wish to view and press RETURN. The designated character image appears on the grid.

7. Rotate

Select this option to turn the character image on the grid anywhere from 0 to 360 degrees. When you press the Select key, EDFONT displays a circle showing the degree selection. Press the Select key to indicate the begin point, then move the cursor to the desired number of degrees on the circle. Press the Select key again. The amount of rotation you select appears as the highlighted portion of the circle with the number of degrees printed underneath. The current character image appears on the grid in the designated position of rotation.

8. Resize

Pressing the Select key here causes EDFONT to display the x-y coordinates for the current font. To change either or both, delete the old number(s) with the CHAR DEL key, type in the new number(s), and press RETURN. Next, position the cursor on Change Size and press the Select key. The new coordinate(s) are registered. To keep the same coordinates, simply move the cursor out of the pop-up.

9. Save

Select this option to save your font file, on disk, at any point in its creation or modification without exiting the program.

10. Set Origin

When you press the Select key here, a crosshair appears on the grid. By moving the cursor, you can move the crosshair to the location you wish to designate as the origin for the current character. When you press the Select key at the desired location, a small box appears on the grid to indicate the new point of origin. Remove the crosshair by moving the cursor completely off the grid.

11. Character

When you position the cursor here and press the Select key, EDFONT displays a table showing upper- and lowercase letters. To change the selection on the table to special characters and numbers, position the cursor on Special Chars and press the Select key. To regain the letters, position the cursor on Letters and press the Select key. Each time you choose a letter or special character by pressing the Select key, that letter/special character is displayed, actual size, directly under the table.

12. To select a particular character from the table, position the cursor on that character and press the Select key, then move the cursor to Edit This One and press the Select key again. If you select a character from an existing font, EDFONT displays that character on the grid. If you select a character to be created, the cursor moves to its designated point of origin on the grid. EDFONT stores whatever you create, in the font file that you assign it, as the character that you selected it to represent.

13. Undo

Select this option to cancel the last operation you performed. After you press the Select key, EDFONT displays the next to last version of the character.

14. Design Menu

This menu offers the following options:

Pixel

Select this option to build a design one grid square at a time. Position the cursor and press Select at the square you wish to fill.

• Line

This option refers to a series of pixels with a defined beginning and end point. After selecting Line and pressing the Select key, move the cursor to a begin point and press Select. Now as you move the cursor, "rubber banding" allows you to see the potential length of the line. When the line stretches to the desired length, press the Select key. All the grid squares between the two points are then filled.

• Paint

This option allows you to fill a continuous series of squares simply by pressing the Select key at the first square, then moving the cursor. You don't need to press Select after each subsequent square as in Pixel. Pressing Select again turns this option off.

Box

Use this option to draw a box with a series of squares. Set two points on the grid to define the size and shape of the box. The first point is the center and the second point, the radius. After you press Select for the center point, use rubber banding to grow and shrink the box. Press Select at the size box you want. (See items 14 and 15 for additional options.)

• Arc

Use this option to draw an arc with a series of squares. After setting the first point, use rubber banding as a guide for the size and sweep of your arc.

• Circle

This option allows you to draw a circle with a series of squares. Set the center of the circle, then use rubber banding as a guide for the radius. (See items 14 and 15 for additional options.)

15. By pressing the Select key or space bar you can switch this option from On to Invert to Off and select any of these by pressing the Select key at the desired option.

On

turns on the pixel.

Invert

turns the pixel from on to off or vice versa, depending on its current state.

Off

turns off the pixel highlight.

16. This option is also switchable, in this case from Filled to Outline.

Filled

fills the space between selected points with pixel squares; available only for Box and Circle.

Outline

fills only a line between selected points with pixel squares.

- 17. This box shows the character/number you are currently designing on the grid in its actual size.
- 18. Col Row

 This box indicates the position of the cursor on the grid (in pixels) at any time.
- 19. This box shows the name of the current font.
- 20. Delete

With this option you can delete the current character from the font along with its image from the grid.

21. Erase

To clear the grid, position the cursor here and press the Select key.

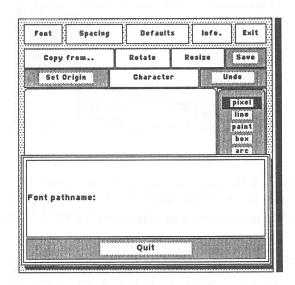
C.4. Using EDFONT

The following exercise is intended to familiarize you with EDFONT. Please follow the instructions below.

1. Invoke EDFONT by typing the following in the Shell input pad.

\$ EDFONT<RETURN>

The EDFONT menu appears.

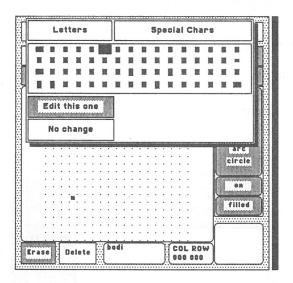


2. At "Font Pathname:" type the following:

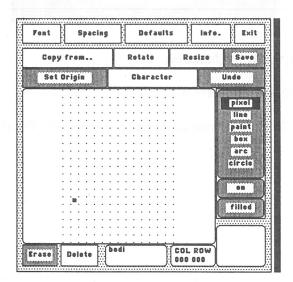
/SYS/DM/FONTS/NEW<RETURN>

3. Move the cursor to Character. Press the Select key to display the alphabet table.

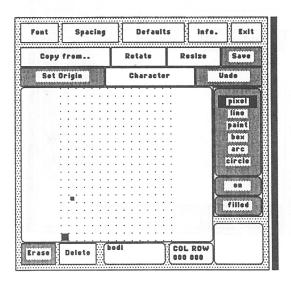
4. Position the cursor on the letter E and press the Select key. The E on the alphabet table is outlined.



5. Position the cursor at Edit This One and press the Select key. EDFONT displays a grid with a small box on the lower left side. This box indicates the default point of origin. To change the letter's point of origin, see **SAMPLE EDFONT DISPLAY**, item 10.

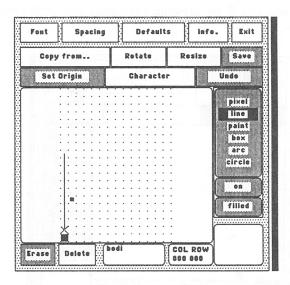


- 6. Check that the Pixel option in the Design Menu (item 13) and the On option (item 14) are both selected. If not, select them.
- 7. Begin constructing your new E by pressing the Select key at the grid square shown in the illustration below.

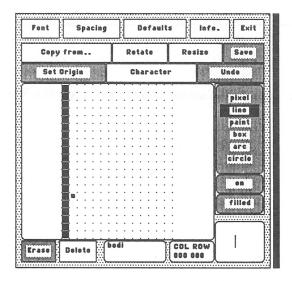


8. At the Design Menu, position the cursor on Line and press the Select key.

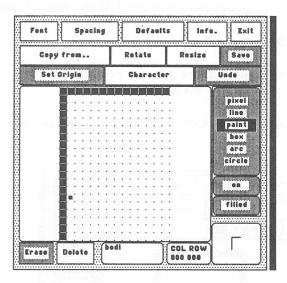
9. Return to the grid and press the Select key at the next box up to be filled. As you move the cursor towards the top of the screen, rubber banding allows you to see the potential length of the line.



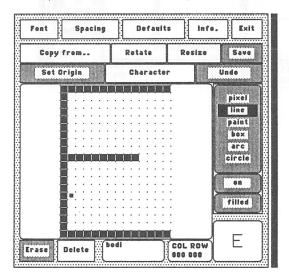
10. Now, move the cursor to the ending box and press the Select key. All squares between these two points automatically fill. You now have an image that looks like this:



- 11. Move the cursor to the Design Menu and select the Paint option.
- 12. Move the cursor to the top square, press the Select key, then move to the right the number of squares shown below. The Paint option allows you to fill in the desired number of squares simply by moving the cursor.



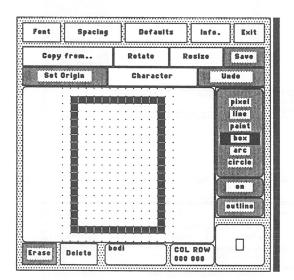
13. Using the Pixel, Line, and/or Paint options, complete the letter "E".



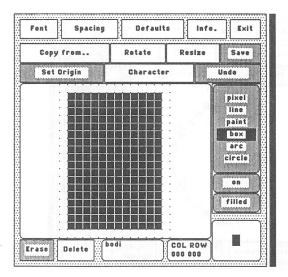
14. Instead of exiting the program, clear the screen for another exercise by moving the cursor to Erase and pressing the Select key.

The next exercise is intended to familiarize you with the remaining EDFONT options.

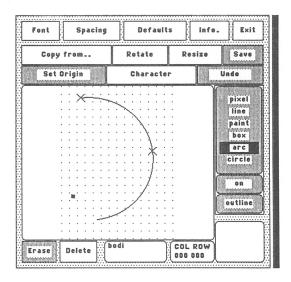
- 15. Check that the On and Outline options are selected. If not, select them.
- 16. Position the cursor on Box and press the Select key.
- 17. On the grid, set up the center of the box by pressing Select. Now, using rubber banding as a guide, select the size of the box.
- 18. When you press the Select key, EDFONT automatically outlines the box.



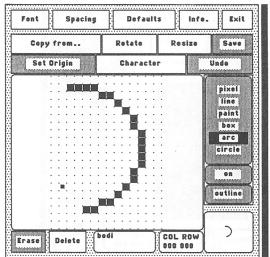
19. Now create a box using the Filled option.



- 20. Use Erase to clear the screen.
- 21. A circle is created in exactly the same way as a box and also has Filled or Outline options. Create a circle using both the Filled and Outline options.
- 22. Use Erase to clear the screen.
- 23. Select Arc and return the cursor to the grid.
- 24. Select two points, pressing the Select key after each one. Rubber banding allows you to see the potential shape of the arc.



25. When you press the Select key a third time, EDFONT draws the arc from point to point.



- 26. To quit, move the cursor to Exit and press the Select key. A pop-up asks if you wish to keep or discard the modifications. Position the cursor on the desired response and press the Select key.
- 27. You have now completed the EDFONT exercise.

C.5. Glossary of Terms

ASCII Value follows: The two-digit hexadecimal number to which a given character corresponds, as

00	28	02	03	04	2 6	97	10	11	12	13	14	15	16	20	21	22	23	25.4	26	27	30	31	32) L	υ n	36	37	40	42	43	44	40	47	50	5 5	5 2	54	55	18	0/
hex	20	02	03	04	26	98	80	09	94	0B	8	8	Q E	3 5	H	12	13	7.4	16	17	18	19	A	3 6	i 5	H	片	20	22	23	24	3 5	27	28	29	3 8	38	B	Œ	1
PK	TO NO.	XIX	XIS	EQI	SNO SNE	B	BS	H	NL (LF)	Ā	E.	В	RRS	858	NOX	馬	XOFF	NE X	NYS	EIB	CAN	M	SUB	3 5	2 2	25 8	SU	SP	3	100	· co	di di	- R	^	-	+ ×	7	1 .	. •	
dec	28	02	: 03	04	R G	97	08	09	10	11	12	13	14	16	17	18	19	20	22	23	24	25	26	72	200	30	31	32	2 3	45	36	30/	39	40	41	42	44	45	46	4/
00	60	62	63	64	A 65	9	70	71	72	73	74	75	76	100	101	102	103	105	106	107	110	111	112	11.5	115	116	117	120	122	123	124	27.	127	130	131	132	134	135	136	10/
hex	30	32	. ω	ω 4	ب ا ا ا	37	38	39	3A	3B	30	3 8	ب ا	40	41	42	43	44	46	47	48	49	A A	ל ל	ל ל	4	4F	50	2.5	53	1 01	7 0	57	58	59	5 X	2 8	A	E E	JC.
r	10	2	. w	4	ח ת	7	00	9	••	~•	^	11	, ~	.,	A	œ	0	7 5	ופי	G	H	H	4	1 >	E t	Z 3	0	סי פ	Z IC	S	Η3	; c	€ <	×	×	- 10	/-	- ,)	1
dec	4 4 8	50	51	52	<u>r</u> 5	55	56	57	58	59	60	61	62	600	65	66	67	6 6	70	71	72	73	74	7 /	76	78	79	80	82	83	84	2 0	87	88	89	2 9	9 4	93	94	CF
<u>p</u>	140	142	143	144	145	147	150	151	152	153	154	155	156	160	161	162	163	165	166	167	170	171	172	17.5	175	176	177													
bex	60	62	63	64	600	67	68	69	64	6В	60	8	6E	70	71	72	73	75	76	77	78	79	7A	7 6	d >	7E	7F													
r	D	σ	C	α	ተወ	ΩF	של	.	٠.	*	۳	Ħ	ם	3 0	Ω η	н,	ß	. 4	< 1	¥	×	Y	N	- ~	~ -	1-	DEL													
dec	96	98	99	100	101	103	104	105	106	107	108	109	011	112	113	114	115	117	118	119	120	121	122	17.	124	126	127													

Character Descriptor

That part of a font which contains the information necessary for the system to display a character. This information is in the form of five scalar components: up, down, right, left, and width. The first four describe how many pixels the character image occupies in two directions. The fifth, width, is not related to device used by EDFONT to tell the system how much space to leave between the origin of one character and the origin of the next. the width of a character image as described in the font header, but is a spacing

Character Image The graphic image stored in memory by the bitmap and displayed on the screen as an array of pixels.

Default Height A value that designates the height (number of pixels occupied by a character image in the vertical direction) of any character in a font file edited by EDFONT. You can change this value.

Default Width A value that designates the width (number of pixels occupied by a character image in the horizontal direction) of any character in a font file edited by EDFONT. You can change this value.

Fixed Width A font with characters that are all of the same width. These fonts produce displays and documents that are monospaced (no allowance for character width differences). Fixed width fonts are most useful whenever column alignment is important.

Font File A file that holds one font. All of the characters that appear on a display are read from a font file. The EDFONT current font file format restricts the character image size to a maximum of 224 by 224 pixels, and the character descriptor components to a range of -127 to 127.

Font Header

That part of a font that contains data for the entire font; namely, the number of characters in the font, the maximum height of characters, the maximum width of characters, the amount of space between characters, the amount of space between lines, and the amount of space that will be skipped for an undefined character. All, except the number of characters, are expressed in pixels. The height and width of characters may vary, but no character can be higher or wider than the values listed in the font header for height and width.

Font Pathname The path that the system must take to find the location of a specific font file. Entered at the initial EDFONT display, the font pathname conforms to the DOMAIN pathname format.

Font

The data compiled by EDFONT relating to a character image created on the grid. A font can consist of as little as one character, but the term usually refers to a group of characters related by size and/or typeface design.

Horizontal Spacing

The amount of space between each character in a font that EDFONT will skip in a horizontal direction. This is user-modifiable data, expressed in pixels, that affects the font as a whole, rather than only one character at a time. (Refer to Figure C-1, item 2.)

Maximum Height The value, in pixels, for the tallest character in a font.

Maximum Width The value, in pixels, for the widest character in a font.

Monospaced Characters

See Fixed Width.

Origin

The x-y coordinate pair set by the system at which a character is displayed, where x is defined as a pixel having a "right" scalar value of 1, and y as a pixel having an "up" scalar value of 1.

Pixel

Any of the tiny picture elements that form a digitalized picture on a tv or crt screen. Pixel also describes the units of space (grid squares) that you use in EDFONT to create or edit a character. Because display pixels are too small to work with comfortably, the EDFONT grid uses enlarged pixels. However, no matter what size pixel you work with, the pixel size on the printed output and on displays other than the grid are set by the particular device you are using and are not modifiable.

Proportionally Spaced Characters
See Varying Width.

Scalar Components

See Character Descriptor.

EDFONT

Space Size

The number of pixels to skip in the horizontal direction when an undefined character is written. When you press a key for a character not in the font, or an arrow key, the cursor moves this number. You can modify this data. (Refer to Figure C-1, item 2.)

Varying Width

A font in which character size varies from character to character. These fonts produce displays and documents that are proportionally spaced (with allowance for character width differences). Varying width fonts are most useful in typesetting and other document preparation applications.

Vertical Spacing

The amount of space between each line of characters that EDFONT will skip. This is user-modifiable data, expressed in pixels, that affects the font as a whole, rather than only one character at a time. (Refer to Figure C-1, item 2.)

Appendix D INVOL

D.1. Background

INVOL initializes physical disk volumes, creates logical volumes, and maintains badspot lists. Once initialized, a volume can be mounted with the MTVOL (MOUNT_VOLUME) command, or can be used to bootstrap the operating system, providing it contains the necessary files.

Disks that have been corrupted by system crashes or network failure can be salvaged without having to be reinitialized, thus saving existing data. See Appendix E for details on SALVOL (SALVAGE_VOLUME).

Logical Volumes

Various logical organizations of a disk are shown in Figure D-1. Each disk has a physical volume label, created when the disk is initialized. Following the physical volume label are one or more logical volumes. Each logical volume is a logically independent storage area and has a name. A logical volume can span all or part of a physical disk. INVOL can create logical volumes when it first initializes the disk, and can add logical volumes later, by partially initializing a physical volume.

Space on the disk that is not allocated to a logical volume is called a "vacancy." For example, on a partially initialized disk, a vacancy follows the logical volumes. After deletion of a logical volume, a vacancy takes its place. If adjacent logical volumes are deleted, one vacancy is formed. When initializing a partial physical volume, you can fill any vacancy with one or more logical volumes.

Badspots

Badspots are defective blocks that cannot be used for data storage. During manufacturing, we write a list of badspots onto every Winchester disk. Normally, you need not alter the factory-supplied badspot list. However, if you determine that the list is incomplete or inaccurate -- as the result of repeated disk I/O errors at the same location -- you can add new badspots to the list (see operation nine at the end of this appendix).

Floppy disks typically do not have factory badspots. In cases of recurrent floppy disk I/O errors, the simplest solution is to replace the floppy disk.

Examples 1, 2, and 3 in Figure D-1 show three possible disk organizations. In Example 1, a single logical volume occupies the entire disk. Example 2 shows a partially initialized disk where a vacancy follows the logical volumes. Example 3 shows a disk on which one or more logical volumes have been deleted, and vacancies thereby created.

D.2. Invoking INVOL

INVOL can run on-line under the AEGIS Shell or can run as a stand-alone utility in the Mnemonic Debugger. To invoke the program from the Shell, type:

D-1 INVOL

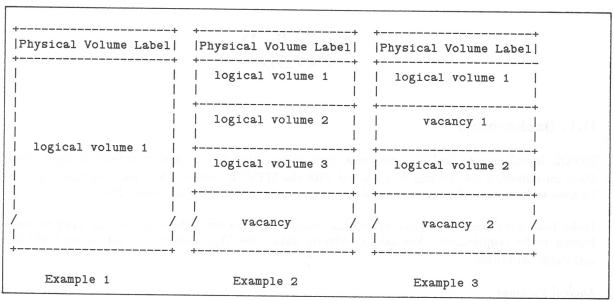


Figure D-1. Sample Logical Disk Organizations

\$ INVOL

From the Mnemonic Debugger, type

> EX INVOL

In either case, the program prompts you for all required information. After gathering the required information, INVOL performs the requested operation.

Exiting

INVOL exits normally when you signal completion of a particular operation. To exit from INVOL before an operation is complete, type CTRL/Z or CTRL/Q (under AEGIS) or CTRL/<RETURN> (under the Mnemonic Debugger). Be careful, though. If you exit while INVOL is initializing the disk, the logical volume being initialized will be unusable. If INVOL is initializing a new physical volume, the entire volume is unusable and will still require initialization before it can be used.

D.3. Operations

INVOL can perform ten operations:

- 1. Initialize a virgin physical volume.
- 2. Partially initialize a physical volume, preserving existing logical volumes.
- 3. Reinitialize a logical volume.
- 4. Delete a logical volume.
- 5. List logical volumes.
- 6. List badspots on a physical or logical volume.

- 7. Input and record badspot information.
- 8. Create or modify an os paging file on an existing logical volume.
- 9. Add to existing badspot list.
- 10. Set or display the sector interleave factor.

The following sections describe these operations.

1. Initializing a Virgin Physical Volume

Every new disk must be initialized before it can be used. When you initialize a new disk, all existing data on the disk are overwritten. Do not initialize a disk that contains any data you need to save. We initialize Winchester disks during the manufacturing process, before installing the system software.

To initialize a new disk, follow this procedure:

- a. INVOL asks which operation to perform. Type 7 to create or replace the badspot list. (See "Recording Badspot Information"). Otherwise, type 1 to initialize a new physical volume.
- b. Specify the type of disk to initialize. INVOL prompts with:

Disk types are:

- W Winchester
- S Storage module
- F 2-sided double density floppy

Respond by typing W, S, or F. To specify a unit number, append 0 or 1 to the letter. For example, S1 denotes storage module unit 1. Unit 0 is the default.

- c. Choose one of the following verification options:
 - 1 No verification
 - 2 Write all blocks on the volume
 - 3 Write and reread all blocks on the volume

If you choose no verification (option 1), INVOL does not read or write to the disk, except to create the volume structure. This option is the fastest, but means that the disk is not verified until it is mounted and read or written by AEGIS.

If you choose the second option, INVOL attempts to write to each block on the disk. The third option, writing and rereading all blocks on the volume, is the safest but also the slowest. For example, to format a complete 33MB Winchester, option 1 requires about five minutes, option 2 requires about fifteen minutes, and option 3 requires about 30 minutes.

A Note on Floppies

If a floppy disk is initialized with INVOL on a busy node, there is a small chance that a format operation will fail, but that the failure will not be reported

to INVOL. For this reason, INVOL writes each block during floppy initialization, even for verification Option 1. If a write fails after a purportedly successful format, INVOL will print the message:

format failed for daddr <disk_address>: <write status> -- retrying format

and will reformat (and rewrite) the track in error. This happens whether or not the floppy has been previously initialized.

d. Enter the average file size, when prompted:

Expected average file size, in blocks (CR for default, 5 blocks):

Press <RETURN> to accept the default value of 5 blocks. Supplying a relatively accurate value for the average file size can save space on the disk, because the volume table of contents (a system table) will be allocated more efficiently.

e. INVOL requests the size (in blocks) and name of each logical volume to be created. After each entry, INVOL tells you how much space remains. After entering the size and name of all logical volumes, enter a blank line to terminate input. A physical volume can contain up to 10 logical volumes.

For example:

There are 1231 blocks available.

volume 1: 1231, vol1

The logical volume size must be at least 30 blocks, and must be a multiple of the track size for the disk. A Winchester disk or storage module has a track size of 18, and a floppy disk has a track size of 8. If you specify a logical volume size that is not a multiple of the track size, INVOL rounds it up to the next multiple track size, and informs you. Note that the physical volume label occupies the first block on the volume. Thus, the size of the first logical volume is always one less than a multiple of the track size.

Logical volume names are optional, and are used only when INVOL lists the logical volumes on the disk (option 5). You cannot change the name of a logical volume after creating it.

f. INVOL requests badspot information by asking whether or not you wish to use the prerecorded badspot list shipped with the disk. ANSWER "YES". If you want to initialize the physical badspot list on a virgin disk, use operation seven, not operation one, to preserve disk integrity. INVOL has retained the badspot prompt in operation one only for compatibility with existing Shell files. After your affirmative response, INVOL displays the badspot list, indicating the physical disk address, cylinder, head, sector, and byte offset range.

If, in later operations, you wish to provide your own badspot information, you can enter badspots for a Winchester disk or storage module in two forms: as hexadecimal physical disk addresses, or as physical cylinder/head/byte

addresses, in the form cylinder-head byte. Terminate badspot entry with a blank line.

For a floppy disk, enter badspots as hexadecimal physical disk addresses, one per line. Terminate badspot entry with a blank line.

- g. INVOL asks for the name of the physical volume. Then it initializes the disk. As formatting proceeds, INVOL displays milestone messages to report its progress. It also displays a message for each volume it initializes, and another when it completes.
- h. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

2. Partially Initializing a Volume

Using operation 2, you can partially initialize a volume, that is, add logical volumes to a physical volume, while preserving the existing logical volumes. Follow this procedure:

- a. INVOL asks which operation to perform. Type 2 to partially initialize a disk.
- b. Specify the type of disk to initialize. (See operation 1b.)
- c. INVOL prints a list of the logical volumes and vacancies on the disk. If the disk has more than one vacancy, INVOL asks where to place the new logical volume by requesting a vacancy number. Indicate the vacancy that you want INVOL to use by entering its number. If there are logical volumes following the vacancy that you choose, INVOL prints a warning message and then automatically increments the volume numbers of those succeeding volumes by one.
- d. Choose a verification option for the logical volume being initialized. (See operation 1c.)
- e. Enter the expected average file size, in blocks. (See operation 1d.) Press <RETURN> for the default value, 5 blocks.
- f. Enter the name and size of each logical volume to be formatted. (See operation 1e.) After each specification, INVOL informs you of how much space is available. Terminate input with a blank line. A physical volume may have up to ten logical volumes.
- g. Enter badspot information. (See operation 1f.) Terminate badspot entry with a blank line.
- h. Enter the name of the physical volume. (See operation 1g.)
- i. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

3. Reinitializing a Logical Volume

You can reinitialize a logical volume, retaining its size and name, with operation 3. All existing data in the volume will be lost. This operation is useful for reinitializing floppy disks, where one logical volume typically occupies the entire physical volume.

To reinitialize a single logical volume, use the following procedure:

- a. INVOL asks which operation to perform. Type 3 to reinitialize a logical volume.
- b. Specify the type of disk to initialize. (See operation 1b.)
- c. Choose a verification option: no verification, write all blocks, or write and reread all blocks. (See operation 1c.)
- d. Enter the expected average file size, in blocks. (See operation 1d.) Press <RETURN> for the default value, 5 blocks.
- e. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

4. Deleting a Logical Volume

To delete a logical volume, use the following procedure:

- a. INVOL asks which operation to perform. Type 4 to delete a logical volume.
- b. Specify the type of disk from which the volume will be deleted. (See operation 1b.)
- c. Enter the number of the logical volume to delete. You can determine the logical volume numbers present on a disk with operation 5.
- d. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

NOTE: INVOL renumbers the logical volumes following the deleted volume.

5. Listing Logical Volumes

To list the logical volumes on a disk, use the following procedure:

- a. INVOL asks which operation to perform. Type 5 to list the logical volumes on a disk.
- b. Specify the type of disk. (See operation 1b.) INVOL lists the volumes on that disk.
- c. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

6. Listing Badspots

To list the badspots in one or more logical volumes, or for the physical volume, use the following procedure:

- a. INVOL asks which operation to perform. Type 6 to list badspots.
- b. Specify the type of disk. (See operation 1b.)
- c. Specify the badspots to be listed, by entering one of the following:

a logical volume number ALL PHYS

Enter a logical volume number to list the badspots in that logical volume. (The logical volumes present on a disk can be listed using operation 5.) Enter ALL to list the badspots in all logical volumes. Enter PHYS to list all badspots on the disk.

d. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

7. Recording Badspot Information

Using operation 7, you can create or replace the badspot list on the disk. (Use operation 9 to add badspots to an existing badspot list.)

- a. INVOL asks which operation to perform. Type 7 to enter the disk's badspot list.
- b. Enter the location of the badspots, one per line. (See operation 1f for the proper format.) Terminate badspot information with a blank line.
- c. After you have typed in the list, INVOL asks you to check for errors. If you made any errors in the list, you must retype the entire list by returning to step a and beginning again.
- d. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

8. Creating or Modifying an OS Paging File on an Existing Logical Volume

You can create an operating system (OS) file or modify the size of an existing one. The OS paging file is required if you intend to run the AEGIS operating system off of this logical volume.

To create or modify an OS paging file, perform the following steps:

- a. INVOL asks which operation to perform. Type 8.
- b. Specify disk type. (See operation 1b.)
- c. Specify logical volume number. The logical volumes present on a disk may be listed using operation 5.

- d. If an OS paging file already exists on this volume, INVOL displays the file's current size and asks if you want to change it. If you reply Y, INVOL proceeds to step e. If you reply N, INVOL skips to step f.
- e. INVOL prompts you to enter the number of pages you want in the OS paging file. Press <RETURN> to use the default, 352 pages. Type "0" (zero) to delete an existing paging file, or specify any number of pages between 1 and 288. If the size you enter is larger than the current OS paging file, INVOL displays milestones as it initializes new disk records.
- f. INVOL asks if you have any more requests. Type Y to return to step a, or type N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

9. Adding Badspot Information

Using operation 9, you can add to the disk's existing badspot list. Run SALVOL when you complete this option.

- a. INVOL asks which operation to perform. Type 9 to add to the disk's badspot list.
- b. Enter the location of the badspots, one per line. (See operation 1f for the proper format.) Terminate badspot information with a blank line.
- c. After you have typed in the list, INVOL asks you to check for errors. If you made any errors in the list, you must retype the entire list by returning to step a and beginning again.
- d. INVOL asks if you have any more requests. Type Y to return to step a, or N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

10. Setting or Displaying the Sector Interleave Factor

Using operation 10, you can set or display the sector interleave factor for a volume. The correct interleave factor is set when a logical volume is created. However, as performance improvements are made, it may become necessary to change it to achieve optimal block read/write rates. Operation 10 displays the current value and the optimal value which we recommend.

- a. INVOL asks which operation to perform. Type 10 to set or display the sector interleave factor.
- b. Specify disk type. (See operation 1b.)
- c. INVOL displays a list of logical volumes for that physical volume. Specify the appropriate logical volume number. INVOL then displays the current sector interleave factor and the value which we recommend.
- d. INVOL prompts for the new interleave factor. If you do not wish to change the interleave factor, enter a carriage return.
- e. INVOL asks if you have any more requests. Type Y to return to step a, or type

N to return to the calling program (AEGIS command Shell or Mnemonic Debugger).

Appendix E SALVOL

E.1. Background

Each logical volume is divided into disk blocks. (See Appendix D, INVOL, for a detailed description of logical disk volumes.) SALVOL verifies, and if necessary, corrects the tables that describe the allocation of disk blocks to the files stored on the disk. SALVOL also returns all blocks that are no longer in use to the free space pool. (Those blocks allocated to temporary files or to deleted portions of permanent files are included).

The label of every logical volume on every physical disk contains its last mount and dismount times. When the Mnemonic Debugger loads AEGIS into memory, it checks these two times. A system crash or improper dismount may mean that the volume has been mounted since its last proper dismount. If so, AEGIS warns you that the volume needs salvaging, and then proceeds to run SALVOL automatically, provided that the NORMAL/SERVICE switch is set at NORMAL.

Even though SALVOL is not run automatically when your node is in SERVICE mode, you still receive a warning message if some boot volume error is detected. Although you can proceed after AEGIS's "Disk needs salvaging" message, we strongly encourage you to take the time to salvage the logical volume. As part of its normal operation, AEGIS periodically updates the essential data on each disk volume. When the volume is wrested from the system's control by either an improper dismount or a system crash, the information on that volume is at most a few minutes out of date. Proper salvaging can identify and correct almost all the inconsistent block allocation information with no loss of files. Of course, memory-resident data buffers that were not written to the volume cannot be reconstructed.

If you mount the volume without running SALVOL, AEGIS incorrectly assumes that the block allocation information is correct. Extensive damage may occur to files that would otherwise be intact. The need to salvage a volume is satisfied only by using SALVOL on that volume.

E.2. Invoking SALVOL

SALVOL can run under the AEGIS operating system or as a stand-alone utility. To invoke the program under the operating system, type:

\$ SALVOL

If you invoke SALVOL under the operating system, it cannot salvage mounted volumes (you must dismount them), nor can it salvage the boot volume on which the operating system is running.

To invoke the program as a stand-alone utility, type the following command to the Mnemonic Debugger:

>EX SALVOL

In either case, SALVOL identifies itself and prompts for the type of disk (Winchester, storage module, or floppy) and the number of the logical volume to be salvaged, as indicated below.

E-1 SALVOL

Salvage volume - Version 6.0

Controller type (W=winchester,S=storage module,F=floppy)? w Salvol options:

-A : read all blocks in all files

-V : verify only (don't write anything to disk)

-U : print uids & vtocx's as well as any filenames

Please input lv_num [-option] . . :

Controller Types

In response to the "Controller type?" prompt, you must specify which storage device you wish to salvage. Do this by typing

type[unit]

where

type

indicates the device type. Valid types are

W for Winchester disks

S for storage modules

F for floppy disks

unit

is an integer indicating the unit number of the device. There must be no blanks between the 'type' and 'unit' specifiers. If this specifier is omitted, the default unit number is 0 (zero). This specifier is useful only when there are two or more of the same storage devices attached to the node (i.e., type "w1" to salvage the second disk in a node with two Winchesters.)

Options

In response to the "Please input lv_num [-option]" prompt, you may specify the number (beginning with 1) of the logical volume you wish to salvage, and one or more of the following three options. (If you omit the logical volume number, logical volume 1 is salvaged.)

-A

Direct the salvager to read every block of every file on the disk. This option is useful because AEGIS cannot always provide detailed descriptive information when an uncorrectable disk read or write error occurs online. SALVOL can diagnose these problems, but does not usually correct their cause.

Without the -A option, SALVOL reads only the minimum number of blocks necessary to ascertain the proper allocation of blocks to files; specifically, it reads the volume-table-of-contents (VTOC), block availability table (BAT), volume label, and file index blocks.

-V

Instruct the salvager to examine the disk for possible inconsistency, but not to correct any damage it finds; do not write anything to the disk. This option allows you to inspect the disk without changing its current state. Select this option only if you believe that salvaging the volume might cause further damage or destroy "evidence" needed to debug or trace a system problem.

Identify problems by UID. Using this option presumes detailed knowledge of the DOMAIN disk structure. It causes the salvager to identify problems not just by filename, but also by UID and VTOC index (VTOCX). The -U option is useful only if you intend to try manually repairing disk problems that SALVOL does not attempt. You should rarely need this option.

Normally, SALVOL collects free blocks, displays a brief summary of free space on the volume, and terminates. If the disk was not properly dismounted, SALVOL may find errors in the block availability table (BAT). Even with the -A option, execution should take at most a few minutes to run on a Winchester disk. The salvager always displays extensive descriptive information about problems, and repairs the volume as necessary, using conservative strategies.

If any file has I/O errors, or if block allocation for a file is detectably wrong (for example, a block is destroyed because some other file wrote over it), SALVOL sets a "file trouble" flag in the file's VTOC entry. Other DOMAIN software (such as the STREAM_\$OPEN system routine) checks this flag upon processing the file. If the flag is set, a warning or fatal error status may be returned.

E.3. Salvaging Strategy

The next two sections describe the salvager's processing strategy, the repairs it can make, and the meanings of the more verbose error messages. Refer to these sections whenever the salvager produces unexpected error messages.

Salvager Message Output

When the salvager processes a disk volume on which the logical structure is intact, it collects free blocks, prints a free block count, sometimes finds that the BAT is bad, and terminates. Just before termination, it confirms that the volume contains no multiply-allocated blocks.

Any other messages -- in particular, messages identifying multiply-defined blocks -- deserve careful attention. The salvager detects and describes four common classes of errors:

- 1. Out-of-range pointers Any block addresses that point outside the logical volume. These are rare, and are usually the result of hardware or operating system errors. When found, the block pointers are set to zero, leaving a "hole" in the file. In addition, if the pointers are file pointers, the file trouble flag is set.
- 2. I/O errors Any read or write attempts that fail at the disk driver level. Most write errors are fatal to the salvager, because it writes only critical blocks that contain space allocation information. For read errors in files, the salvager sets the trouble flag but takes no corrective action, on the assumption that other tools may be able to recover data.
- 3. Block header validation errors These are "soft" I/O errors which, like disk driver errors, are fatal on critical system blocks. In normal files, these errors are simply detected and described, and a file trouble flag is set. Block header validation errors are usually the result of hardware or software failure.
- 4. Multiply-allocated blocks These blocks are allocated to more than one file, or to a file and to a system structure, such as the VTOC, the BAT, or the badspot list. Multiply-allocated blocks may occur if you ignore the AEGIS warning to run the salvager. Less often, they result from AEGIS errors.

The salvager attempts to repair multiply-allocated blocks, and may display a summary of blocks that have been moved on the volume. In addition, SALVOL displays a list of all files for which it set the trouble flag. Note that the trouble flag is set only if the salvager is sure that a file has problems (an out-of-range pointer that was zeroed, an I/O header error, or a multiply-allocated block that was removed from the file map). For example, if the salvager finds a multiply-allocated block that is owned by two files, and can determine which file the block belongs to, then it sets the trouble flag only for the non-owning file.

Finally, the salvager may report on some uncommon conditions, including:

This indicates that the logical structure on the volume is severely damaged.

Mount the volume under AEGIS (but not as the boot volume), and move files to other volumes. If the error persists you may need to reinitialize this volume.

disk full This error prevents the salvager from repairing the disk. Mount the volume under AEGIS (but not as the boot volume) long enough to delete one or more files, then dismount and salvage it.

internal errors These include nonzero multiply-allocated block count, bad linked lists, etc. Most are fatal to the salvager, and may mean that the volume itself is in trouble.

A number of detailed errors are associated with system blocks, including:

no boot file Every logical volume from which AEGIS is started must have a boot file. Hence, this error can be fixed only by reinitializing the volume. If this logical volume is not used to start the operating system, ignore the message.

trouble with badspot list

Usually, the same block address appears twice in the badspot list; just ignore it. It could mean that the badspot list points to a block in the BAT. Ignore the message unless a BAT block really is bad (signified by repeated I/O errors), in which case the volume must be reinitialized.

vtoce with incorrectly hashed uid found

The VTOCE (VTOC entry) is unusable, so respond "Y" to the salvager's "delete it?" prompt, unless you want to save the VTOCE to trace a probable AEGIS error.

too many bad spots

Table overflow occurred in salvager. This is okay unless multiply-allocated blocks are found on the volume, in which case repair logic may fail.

label check error Apparently, some other process is writing to the volume while the salvager is running; this error should never happen.

badspot list in ly label is full

System limitation exceeded. Either the badspot list has been misused or the physical disk must be replaced.

vtoc chain pointer zeroed

Some files may have been lost because VTOC chains are used to handle overflow of files hashed (by UID) to a full block.

Salvager Processing Strategy

This section describes some subtleties of salvager processing. This information is not needed for typical use of the salvager. In extreme cases, however, the details provided below may be helpful.

Two high-level observations first: DOMAIN disk volumes are structured so that naming directories and space/location information (in a VTOC) about files are kept separately. Currently, the salvager does not synchronize these on-disk structures. That is, it understands the UID file system and VTOC, but not naming directories. In particular, it cannot detect orphans, that is, files in the VTOC that have no names. Second, disk blocks on floppy disks are not self-identifying, as they are on the Winchester. Hence, the salvager's diagnosis and repair logic is much more limited with floppy disks than with Winchester disks.

The salvager is a 2-pass processor, but the second pass is executed only if multiply-allocated blocks are detected. During the second pass, the salvager completes the list of owners of each multiply-allocated block, if, as in most cases, this cannot be done in the first pass. Both passes consist of a sequential VTOC scan, including descent of any VTOC block chains which result from (UID) hash overflow. For each VTOC entry that points to a permanent file, SALVOL reads the file map or list of block addresses sequentially to construct a new block availability table.

SALVOL reads and verifies the file blocks themselves only if you choose the -A option, or if problems must be resolved. Pass two terminates when the owners of all multiply-allocated blocks are identified, and thus may not be a complete sequential pass. After each pass, SALVOL repairs the volume if multiply-allocated blocks were found, and writes out the updated BAT and logical volume label.

I/O errors that occur on physical and logical volume labels or on the block availability table (BAT) for a logical volume are fatal to the salvager. All other errors are reported, but are non-fatal.

The salvager corrects I/O errors in files only in the following special case. AEGIS may report an I/O error in a file if the block header is incorrect. If the block is multiply-allocated, which could cause such an error report, the salvager allocates it exclusively to the "real" owner, and zeros the pointer(s) in any other file(s), leaving holes but eliminating the I/O errors. On a floppy, because the owner cannot be determined, SALVOL copies the block into all "owner" files, and sets the trouble flag. After the salvager terminates, you can inspect the data to determine which files are really intact.

SALVOL zeros bad block pointers. This loses data, and even whole files when the bad pointers are in the VTOC itself, but currently the salvager cannot locate the correct data even if it is on the disk.

Generally, the salvager always repairs the BAT (except in case of hard I/O errors) and the VTOC. Thus, if AEGIS badly malfunctions, writing normal file blocks over the BAT or VTOC blocks, for example, the salvager repairs the BAT or VTOC and file. To do so, it copies the data into a newly allocated block and reinitializes the overwritten block.

If a block is multiply-allocated to both the badspot list and to a file or a VTOC chain, the salvager tries to copy any potentially valid data to a newly allocated block. If the block is in the badspot list because of persistent device level errors, however, the copy may fail; the salvager then prompts for alternatives. The salvager and badspot listing cannot be used to correct persistent errors in the BAT or VTOC hash space, however. The salvager aborts in the former case, and simply reports the I/O error in the second case. The only solution is to reinitialize the volume around such badspots using INVOL.

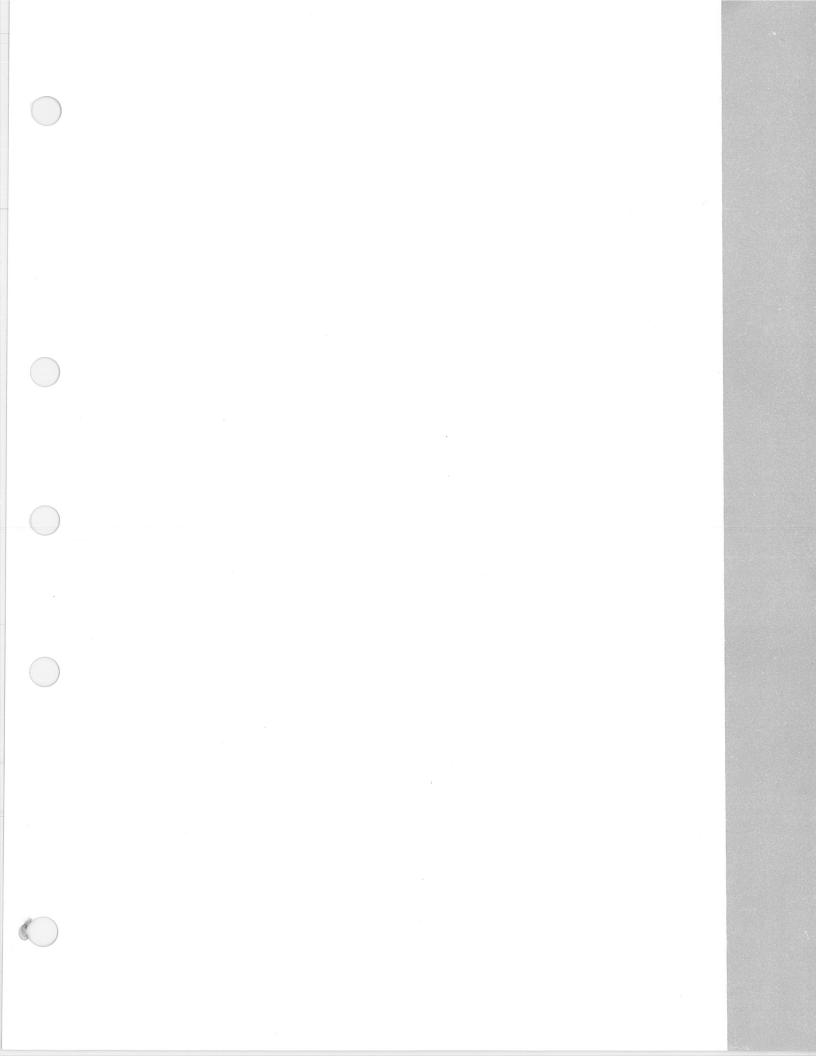
E-5 SALVOL

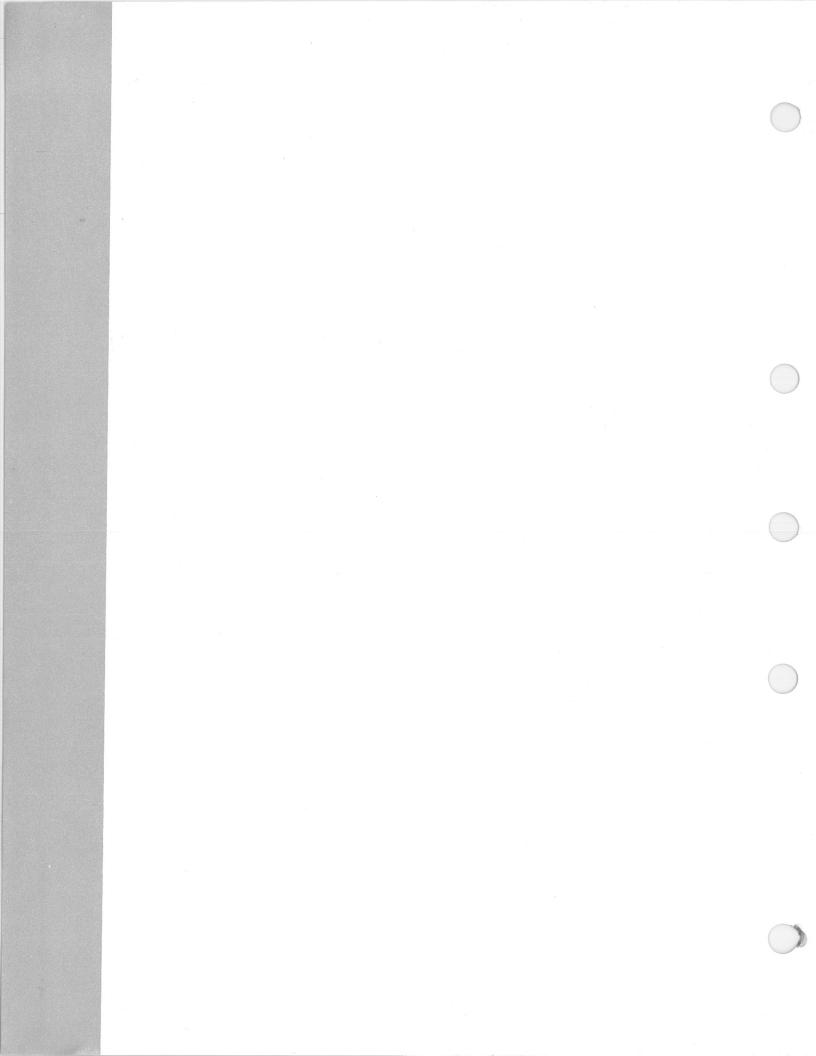
Finally, note that table overflow in the salvager is most serious. You can try to rerun the salvager several times, if some repair is apparent on each run, and if the salvager attempts to complete in each case without doing additional damage. However, in some cases, data may be unnecessarily lost. You should never mount the volume under AEGIS while this condition persists, except to copy/dump and delete files, or to reinitialize the volume.

E.4. Limitations

- SALVOL can process only one logical volume at a time.
- SALVOL can correctly process at most 64 multiply-allocated blocks and/or header validation errors on a logical volume.
- Because flop y disks do not have block headers, so that blocks are not self-identifying, the salvager is much less effective in repairing floppy disk problems than it is in repairing Winchester disk or storage module problems.

E-6





Index

AA 2-2	4-228
Abort severity 4-2, 4-229	AU 2-9
Abort text search 2-3, 2-66	Autohold mode, window 2-81
ABRT 2-3	D 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Absolute mode 4-277	Background color, display 2-10
Absolute/relative mode 4-278	Background process output 4-17, 4-18
ABTSEV 4-2	Backup history 4-294
Access control 4-4, 4-88, 4-238	Badspots, described D-1
Access rights 4-92	BGC 2-10
Accounts 4-85	See also INV
Acknowledge DM alarms 2-2, 2-6	BIND 4-10
ACL 4-4	Bind object modules 4-10
ACLs	BLDT 4-16
display or copy 4-4	BOFF 4-17
edit 4=88	BON 4-18
explained 4-91	Boot Shell 2-36
salvage 4-238	CALENDAR 4-19, A-1
AD 2-4	Cancel ECHOing 2-66
Add to a window group 2-87	Cancel text search 2-3, 2-66
Address space, list 4-160	Carriage control 4-202, 4-205
AL 2-5	CASE 2=11
Alarm server 4-245	change in text 2-11, 4-275
Alarms	CASE construct 4=243
acknowledge 2=2	Case sensitivity 2=63
pop window 2=6	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Catalog a node 4-55
send 4=245	Catalog an object 4-58
Analyze network performance 4-190,	Catalog uncataloged objects 4-126
4=193, 4=213	Catenate files 4-20
AP 2-6	CATF 4-20
AR 2=7	CC 2-12
ARCF 4-7	CE 2-13
Archive files 4-7	Change
ARGS 4-9	case of text 2-11, 4-275
Arrow keys	home directory 2-47, 4-21
move down one line 2-4	icon into window 2-39
move left one character 2-5	object name 4-22
move right one character 2-7	password 2-46, 4-24
move up one line 2-9	window into icon 2-39
set scale factors 2-8	window position 2-91, 2-92
AS 2-8	window size 2-85, 2-86
ASCII carriage control 4-202, 4-205	Check for file existence 4-121
Assign user input to variables 4-225, 4-227,	Check variable existence 4-122

CHHDIR 4-21	CRD 4-42
CHN 4-22	Create
CHPASS 4-24	directory 4-42
CHPAT 4=25	edit pad 2-13
Close a pad 2-82	file (zero=length) 4-44
Close a window 2-82	link 4-45
CMDF 2-14	local registry 4-49
CMF 4=28	network registry 4-49
CMS 2-15	paste buffer 2-18
CMSRF 4-30	protected subsystem 4-51
CMT 4-31	read-only edit pad 2-22
Command line interpreter 4-250	remote process 4-46
Command line parser 3-3	user account 4-85
standard command options 3-5	user change request 4-52
Command search rules 4-53	window group 2-87
Compare directory trees 4-31	window to transcript pad 4-48
Compare files 4-28, 4-30	Create a process 2-16
Compare strings for equality 4-117	background 2-20
Conditional statement 4-155, 4-243	server 2-21
Continue a process 2-23	CREFS 4-43
Conventions	CRF 4-44
in this manual 2	CRL 4-45
Convert file types 4-59	Cross-reference text strings 4-43
Coordinates	CRP 4-46
X/Y 1-2	CRPAD 4-48
Сору	CRRGY 4-49
directory tree 4-39	CRSUBS 4-51
display image 2-97	CRUCR 4-52
entire display 4-38	CSR 4=53
file 4-34	CTNODE 4-55
link 4-36	CTOB 4-58
system boot file 4-33	Current date and time 4-61
text to paste buffer 2-95	Cursor movement
window 2=12	move down one line 2-4
Count strings in a file 4-128	move left one character 2-5
CP 2-16	move right one character 2-7
CPB 2=18	move up one line 2-9
CPBOOT 4-33	Cut and paste
CPF 4-34	copy image 2-97
CPL 4-36	copy text 2-95
CPO 2-20	cut text 2-96
CPS 2-21	paste text 2-98
CPSCR 4=38	Cut text to paste buffer 2-96
CPT 4=39	CV 2-22
CPU time, show 4-224	CVT REC UASC 4-59

DATE 4-61	Documentation conventions 2
DC 2-23	DQ 2-24
DCALC 4-62	DR 2=25
DEBUG 4-65	DS 2-26
Debug high-level programs 4-65	DSPST 4-76
Declare keyboard type 2-43	Dumb terminal emulator 4-108
Define user names 4-102	ECHO 2=27
Defining keys 2-44	
Defining points and regions 1-1	Echo command line arguments 4-9
Delete	ED 2=28, 4=78
character under cursor 2-28	EDACCT 4-85
directory 4-72	EDACL 4-88
duplicate lines from file 4-69	EDFONT 4-96, C-1
file 4-70	Edit
link 4-71	file (DM) 2-13
preceding character 2-29	file (Shell) 4-78
variables 4=73	font 4-96, 4-201, B-1, C-1
window from window group 2-88	magtape descriptor file 4-97
Desk calculator functions 4=62	network root directory 4-99
Directory	stream 4-105
copy 4=39	EDMTDESC 4-97
create 4-42	EDNS 4-99
delete 4=72	EDPPO 4-102
list contents 4=167	EDSTR 4-105
	EE 2-29
naming 4=189	EEF 2-30
salvage 4=239	EI 2-31
working 4=302	Elapsed time, show 4-224
Disks	EM3270 4=107
dismount 4=74	EMT 4-108
initialize 4-157, D-1	Emulate 3270 terminal 4-107
mount 4=185	Emulate dumb terminal 4-108
salvage 4=241, E=1	EN 2-32
Dismount a volume 4-74	End-of-file marks 2-30
Display	ENSUBS 4-114
refresh 2-59	Enter a protected subsystem 4-114
set background color 2-10	ENV 2=33
Display process status 4-76, 4-217	Environment variables 2-33, 4-124
DLDUPL 4-69	EOF marks 2=30
DLF 4=70	EOFF 4-115
DLL 4-71	EON 4-116
DLT 4-72	EQS 4-117
DLVAR 4-73	ER 2-34
DM commands from a Shell 4-304	Erase marks 2=15
DM messages 2-49	ES 2=35

ESA 4-118 FST 4-147 EX 2-36 GM 2-38 Execute DM script 2-14 Grow a window 2-85, 2-86 EXFLD 4-119 EXISTF 4-121 HELP 4-148 EXISTVAR 4-122 for Shell scripts 4-149 EXIT 4-123 Histogram, program counter 4-151 Exit DM to Boot Shell 2-36 HLPVER 4-149 Exit from a loop 4-123 Hold mode, window 2-89 Expand macro definitions 4-181 Home directory 4-21 EXPORT 4-124 HPC 4-151 External symbol address 4-118 Hysteresis 4-280 Extract fields of data 4-119 ICON 2-39 Fault status 4-147 Icon default positions 2-40 File existence test 4-121 Icons File length 4-128 create 2-39 File protection 4-4, 4-88, 4-238 set default positions 2-40 Find blocks of text 4-142 IDF 2-40 Find spelling errors 4-145 IF 4-155 Initialize a disk volume 4-157, D-1 Find text strings 1-2, 4-139, 4-142 FIND_ORPHANS 4-126 INLIB 4-156 FL 2-37 Insert characters 2-35 FLEN 4-128 Insert mode 2-31 Floating-point error mask 4-144 Install user libraries 4-156 Floppy disks Interrupt a process 2-26 INV 2-42 read backup 4-219 write backup 4-294 See also BGC FMC 4-129 Invisible windows 2-90 FMT 4-130 INVOL 4-157, D-1 Fonts KBD 2-43 described B-1 KD 2-44 edit 4-96, 4-201, B-1, C-1 Key names load 2-37 standard 1-8 FOR 4-137 Key naming conventions 1-8 FOR loop 4-137 Keyboards Format 880 map 1-9 multiple columns 4-129 declare type 2-43 text file 4-130 low-profile map 1-9 FORTRAN carriage control 4-202, 4-205 Keys, defining 2-44 FPAT 4-139 L 2-46 FPATB 4-142 LAMF 4-158 FPPMASK 4-144 Laminate files 4-158 Free disk blocks, list 4-180 Language level debugger 4-65 FSERR 4-145 LAS 4-160

LBR 4-162 return to top 4-197 LCNODE 4-165 WHILE 4-303 LD 4-167 LOPSTR 4-174 Librarian utility 4-162 LRGY 4-175 Libraries, install user 4-156 LTY 4-176 Line mode editor 4-78 LUSR 4-177 LVAR 4-179 Link сору 4-36 LVOLFS 4-180 create 4-45 MACRO 4-181 delete 4-71 Magnetic tape descriptor file 4-97 List Marks 2-25, 2-38, 2-57 address space 4-160 erase 2-15 connected nodes 4-165 go to 2-38 directory contents 4-167 place 2-25 installed types 4-176 replace 2-57 locked objects 4-171 Merge files 4-262 network registry sites 4-175 Messages, DM 2-49 open streams 4-174 Mount a disk 4-185 users logged in 4-177 Mouse characteristics 4-277 variables 4-179 Move a file 4-187 volume free space 4-180 Move a pad 2-50, 2-51, 2-53, 2-54, 2-55 LKOB 4-170 Move a window 2-91, 2-92 LLKOB 4-171 Move cursor LO 2-48 down one line 2-4 Load a font 2-37 left one character 2-5 Locate blocks of text 4-142 right one character 2-7 Locate spelling errors 4-145 to bottom of window 2-67 Locate text strings 1-2, 4-139, 4-142 to DM window 2-68 Lock an object 4-170 to end of line 2-76 Locked objects to front of line 2-72 list 4-171 to input window 2-71 lock 4-170 to next icon 2-75 unlock 4-285 to next tab 2-69 Log in to a node 2-46 to next window 2-74 Log in to a process 4-172 to previous tab 2-70 Log off a node 2-48 to previous window 2-73 Log out 2-48 to top of window 2-78 Logical volumes, described D=1 to window border 2-79 Login 4-172 up one line 2-9 list users 4-177 MSG 2-49 to a node 2-46 MTVOL 4-185 to a process 4-172 Multiple columns 4-129 Loops MVF 4-187 exit from 4-123 Naming directory 4-189 FOR 4-137

Naming server 4-99 PAGF 4-203 ND 4-189 Paginate a file 4-203 Negate a command or expression 4-198 Password 4-24 NETMAIN 4-190 Paste buffers 2-95, 2-96, 2-98 log files 4-191, 4-192 create 2-18 NETMAIN_CHKLOG 4-191 Paste text from buffer 2-98 NETMAIN_NOTE 4-192 PB 2=50 NETSTAT 4-193 PH 2-51 NETSVC 4-195 Pipeline data, save 4-274 Network maintenance 4-190, 4-191, 4-192 Plain mode protocol 4-258 log files 4-191, 4-192 PN 2-52 Network registry 4-85 Points create 4-49 defining 1-1 list sites 4-175 Pop a window 2-93 salvage 4-240 PP 2-53 Network services 4-195 PPON format 4-91, 4-102 Network statistics 4-193, 4-213 PPRI 4-204 NEWLINE, insert 2-32 PRF 4-205 NEXT 4-197 PRFD 4-212 Nibbled mode protocol 4-259 Print a file 4-205, 4-212 Node clock 4-19, A-1 Print server 4-216 Nodes, list connected 4-165 Priority of a process 4-204 NOT 4-198 Probe network 4-213 NS_HELPER 4-99 PROBENET 4-213 Processes Object types 4-199 continue 2= 23 OBTY 4-199 create 2-16 OLD_EDFONT 4-201, B-1 create background 2-20 Open streams, list 4-174 create remote 4-46 Operating system build time 4-16 create server 2-21 Options, standard 3-5 display status 4-76, 4-217 Orphan objects 4-126 priority 4-204 OS 4-202 quit 2-24 Overstrike 4-202 stop 2-24, 4-253 Overstrike mode 2-31 suspend 2-26 Pads Program counter histogram 4-151 close 2-82 Protected subsystems create 2-13, 2-22 create 4-51 move horizontally 2-51 enter 4-114 move to bottom 2-50 execute manager 4-307 move to top 2-54 set/display attributes 4-265 scroll by lines 2-55 PRSVR 4-216 scroll by pages 2-53 PST 4-217 set read/write mode 2-58 PT 2-54 write to a file 2-56 Push a window 2-93

SALD 4-239 PV 2-55 SALRGY 4-240 PW 2-56 Salvage Quit a process 2-24 ACL structure 4-238 directory 4-239 Range of text 1-4 Raw characters 2-34 disk 4-241, E-1 RBAK 4-219 network registry 4-240 RDYM 4-224 SALVOL 4-241, E-1 READ 4-225 Save data in a pipeline 4-274 Read a backup tape 4-219 Save transcript pad 2-52 Read a file 2-22 SC 2-63 Read a floppy disk 4-219 Screen timeout delay 4-242 Read a foreign tape 4-231 Read mode 2-58 execute (DM) 2-14 Read user input 4-225, 4-227, 4-228 Scroll mode, window 2-94 READC 4-227 SCRTO 4-242 READLN 4-228 Search for text Ready message 4-224 abort 2-3 blocks (Shell) 4-142 Receive file from remote host 4-255 Refresh a window 2-60 in files (DM) 1-2 Refresh screen 2-59 in files (Shell) 4-139, 4-142 Regions 2-25 SELECT 4-243 defining 1-2 Send alarm messages 4-245 Send file to remote host 4-257 Regular expressions 1-2, 1-4 set case sensitivity 2-63 SEND ALARM 4-245 SET 4-248 summary 1-7 Relative mode 4-278 Set arrow key scale factors 2-8 Remote transmissions Set search case sensitivity 2-63 receive 4-255 Set tabs 2-77 Set window color 2-42 transmit 4-257 RETURN 4-229 SH 4-250 Return from current Shell level 4-229 Shell Return to top of a loop 4-197 evaluate variables 4-115, 4-116 Reverse lines 4-230 execute 4-250 execute DM command 4-304 REVL 4-230 execute script at current level 4-261 Rights to objects 4-92 execution trace 4-305, 4-306 RM 2-57 RO 2-58 handle background output 4-17, 4-18 RS 2-59 set conditions 4-248 verify command lines 4-289, 4-290 Rubberbanding 2-27, 2-86, 2-92 RW 2-60 Shell command format 3-1 RWMT 4-231 Shell completion status 4-2 Shell conditions 4-248 S 2-61 Shell flags SALACL 4-238 -B 4-17, 4-18

-E 4-115, 4-116 read/write foreign 4-231 -V 4-289, 4-290 write backup 4-294 -X 4-305, 4-306 TB 2-67, 4-269 Show OS version 4-16 TCTL 4-270 Shrink a window 2-85, 2-86 TDM 2-68 SHUT 2-64 TEE 4-274 See also EX Test for file existence 4-121 Shut down system 2-64 Text echoing 2-27 See also EX TH 2-69 SID 4-91 THL 2-70 Signal a process 4-253 TI 2-71 SIGP 4-253 Time zone settings 4-281, A-2 SIO line, configure 4-270 TL 2-72 SIORF 4-255 TLC 4-275 SIOTF 4-257 TLW 2-73 SO 2-65 TN 2-74 Sort files 4-262 TNI 2-75 SOURCE 4-261 Touch pad characteristics 4-277 Special characters (DM) 1-11 TPM 4-277 Special characters (Shell) 3-2 TR 2-76 Spelling errors, find 4-145 Traceback from a fault 4-269 SQ 2-66 Transliterate characters 4-275 SRF 4-262 Transmit file to remote host 4-257 Standard key names 1-8 TS 2-77 Status code translation 4-264 TT 2-78 STCODE 4-264 TWB 2-79 Stop a process 2-24, 4-253 Types, list installed 4-176 Streams, list open 4-174 TZ 4-281 Subject identifier 4-91 UCTNODE 4-283 SUBS 4-265 UCTOB 4-284 Substitute text strings 2-61, 2-65, 4-25 ULKOB 4-285 first occurrence only 2-65 Uncatalog a node 4-283 Suspend a process 2-26 Uncatalog an object 4-284 System calendar 4-19, A-1 UNDO 2-80 System ready message 4-224 Undo DM commands 2-80 Tab left 2-70 Unlock an object 4-285 Tab right 2-69 User accounts 4-85 Tab stop settings 2-77 User Change Requests 4-52 Tabs User libraries 4-156 move left 2-70 Variables move right 2-69 assign values from input 4-225, set 2-77 4-227, 4-228 Tapes change Shell to Environment 4-124 read backup 4-219 check existence 4-122

delete 4-73 Environment 2-33, 4-124 evaluate 4-115, 4-116 list 4-179 VCTL 4-287 VOFF 4-289 Volumes dismount 4-74 mount 4-185 VON 4-290 VSIZE 4-291 VT100 4-292 set terminal characteristics 4-287 set window pane size 4-291 terminal emulator 4-292 WA 2-81 WBAK 4-294 WC 2-82 WD 4-302 WDF 2-84 WG 2-85 WGE 2-86 WGRA 2-87 WGRR 2-88 WH 2-89 WHILE 4-303 WHILE loop 4-303 WI 2-90 Window boundaries 1-3, 2-84 define default 2-84 Window groups create/add to 2-87 delete window from 2-88 Window modes autohold 2-81 hold 2-89 scroll 2-94 visible/invisible 2-90 Windows close 2-82 copy 2-12

copy image 2-97 create 2-13, 2-22

grow and shrink 2-85, 2-86

make icons 2-39 move 2-91, 2-92 pop/push 2-93 refresh single 2-60 set color 2-42 WM 2-91 WME 2-92 Working directory 4-302 WP 2-93 Write a backup tape 4-294 Write a floppy disk 4-294 Write a foreign tape 4-231 Write edit pad to file 2-56 Write files to standard output 4-20 Write mode 2-58 WS 2-94 X/Y coordinates 1-2 XC 2-95 XD 2-96 XDMC 4-304 XI 2-97 XOFF 4-305 XON 4-306 XP 2-98 XSUBS 4-307

READER'S RESPONSE

We use readers' comments in revising and improving our documents.

Document Title: DOMAIN System Command Reference Order Number: 002547 Revision: 03 Date of Publication: July, 1985 What is the best feature of this manual? Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.) What type of user are you? _____ Systems programmer; language _____ Applications programmer; language _____ ____ Manager/Professional ____ Technical Professional _____ Adminstrative/Support Personnel _ Student programmer User with little programming experience ____Other How often do you use your system? Nature of your work on the DOMAIN System: Your name Date Organization Street Address City State Zip/Country No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse), tape, and mail. FOLD



BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 78

CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC. Technical Publications P.O. Box 451 Chelmsford, MA 01824 NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES



READER'S RESPONSE

We use readers' comments in revising and improving our documents. Document Title: DOMAIN System Command Reference Order Number: 002547 Revision: 03 Date of Publication: July, 1985 What is the best feature of this manual? Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.) What type of user are you? Systems programmer; language _____ ____ Applications programmer; language _____ ____ Manager/Professional ____ Technical Professional ____ Adminstrative/Support Personnel ____ Student programmer User with little programming experience Other How often do you use your system? Nature of your work on the DOMAIN System: Your name Date Organization Street Address City Zip/Country State No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse), tape, and mail. FOLD



NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 78

CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC. Technical Publications P.O. Box 451 Chelmsford, MA 01824



READER'S RESPONSE

We use readers' comments in revising and improving our documents.

Document Title: DOMAIN System Command Reference Order Number: 002547 Revision: 03 Date of Publication: July, 1985 What is the best feature of this manual? Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible.) What type of user are you? Systems programmer; language _____ Applications programmer; language Manager/Professional ____ Technical Professional _____ Adminstrative/Support Personnel Student programmer User with little programming experience Other How often do you use your system? Nature of your work on the DOMAIN System: Your name Date Organization Street Address Zip/Country State City No postage necessary if mailed in the U.S. Fold on dotted lines (see reverse), tape, and mail. FOLD



NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 78

CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC. Technical Publications P.O. Box 451 Chelmsford, MA 01824



FOLD